EKS Networking 101

AWS Professional Services

2023.11

Agenda

- Networking foundation
- EKS Control Plane Network
- EKS Data Plane Network (Amazon VPC CNI)
- Ingress Controller

aws

Topics not covered in this session

- Kubernetes Basics
- Linux IPtables
- Mesh Network (App Mesh, Istio)
- Amazon VPC Lattice

aws

Networking foundation

aws

Linux Networking Primitives for Kubernetes

Node
The basis of Kubernetes Workload

Cgroups
Linux kernel feature

Namespace
Process isolation

Network Namespace
Creates a copy of the network stack

Veth
Virtual Ethernet

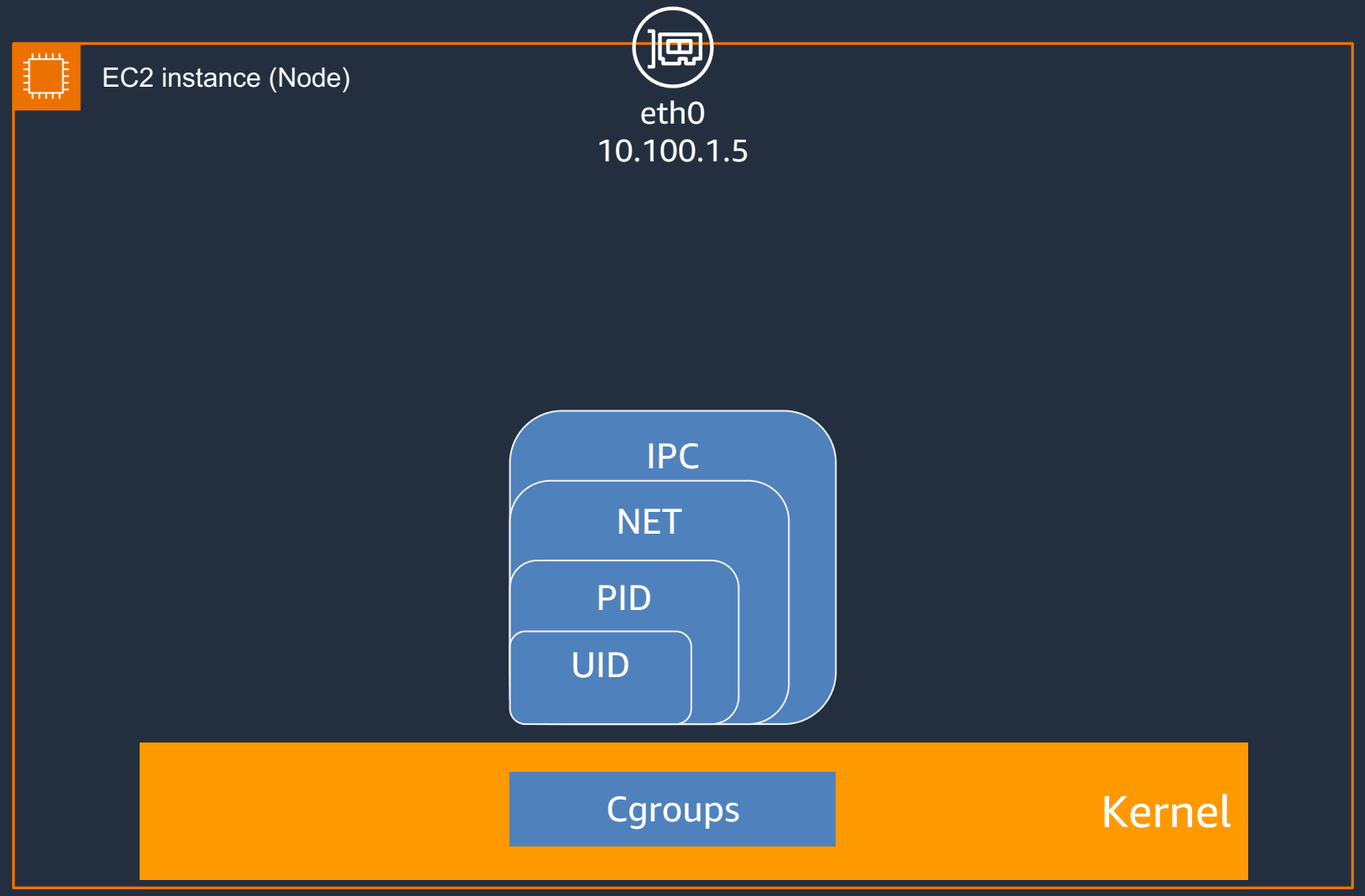
Bridge Interface
Provide connectivity

Pause Container
Manages the namespace for each pod

Kubenet
CNI Plugin

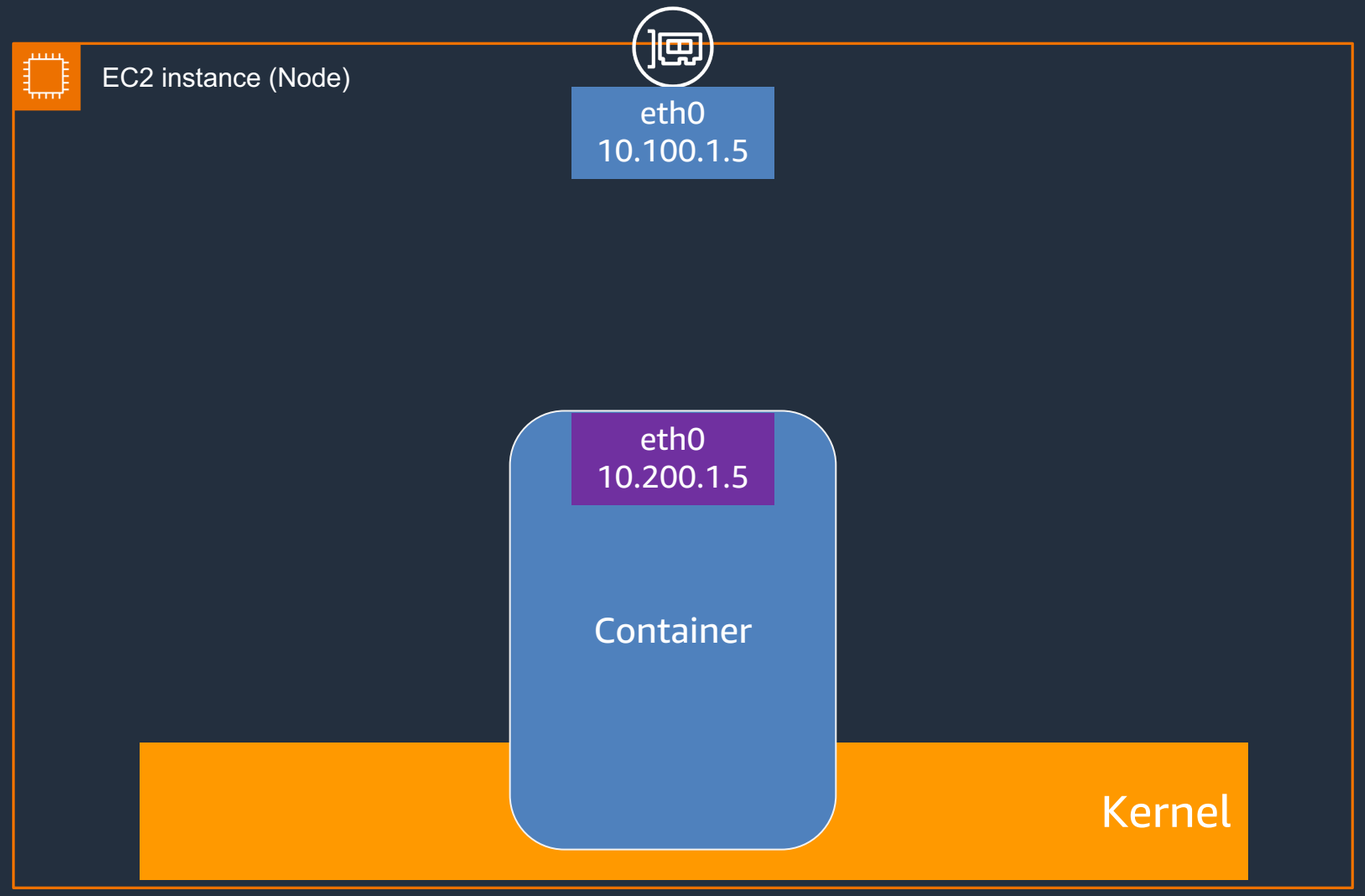
aws

Linux Networking Primitives for Kubernetes



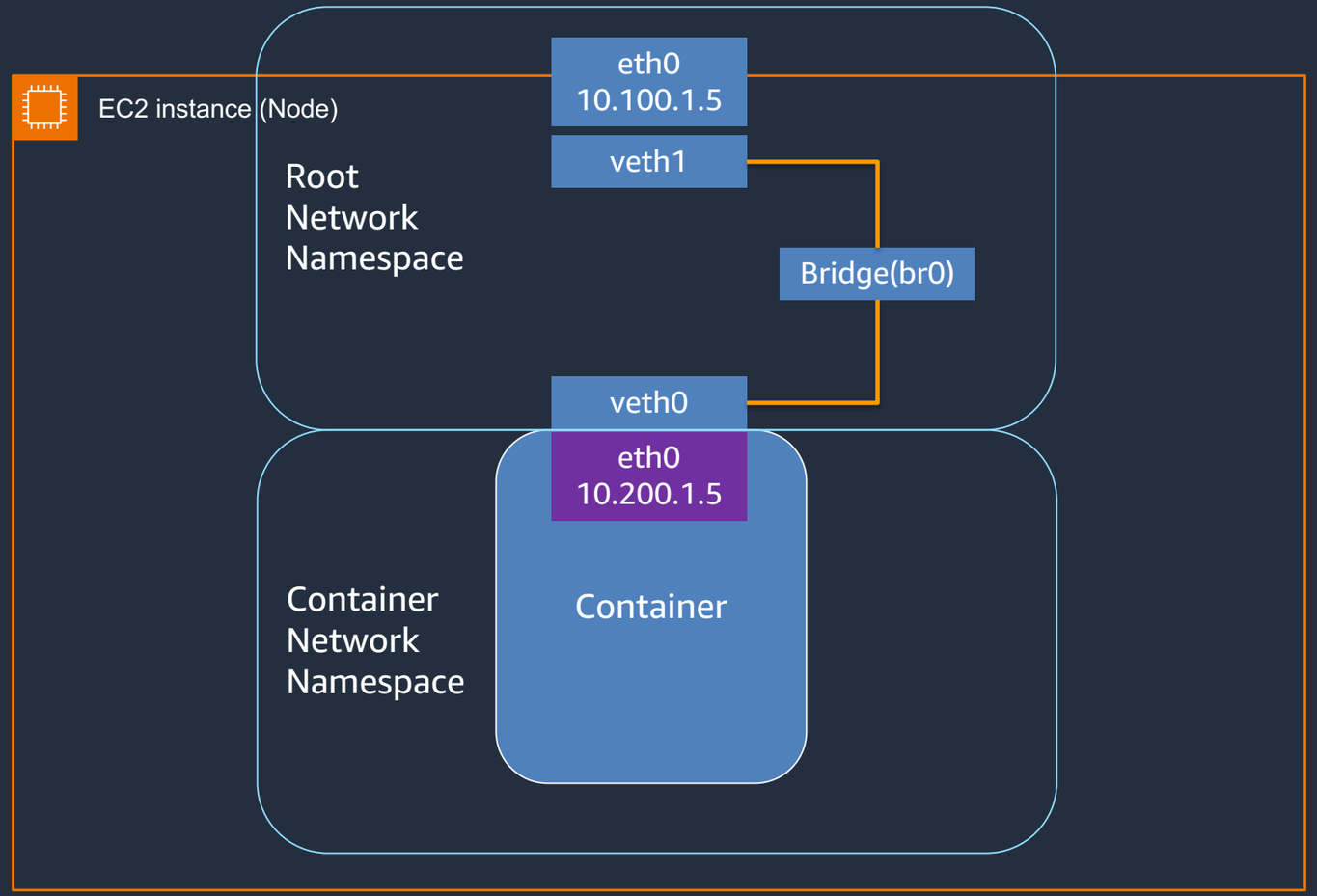
aws

Linux Networking Primitives for Kubernetes



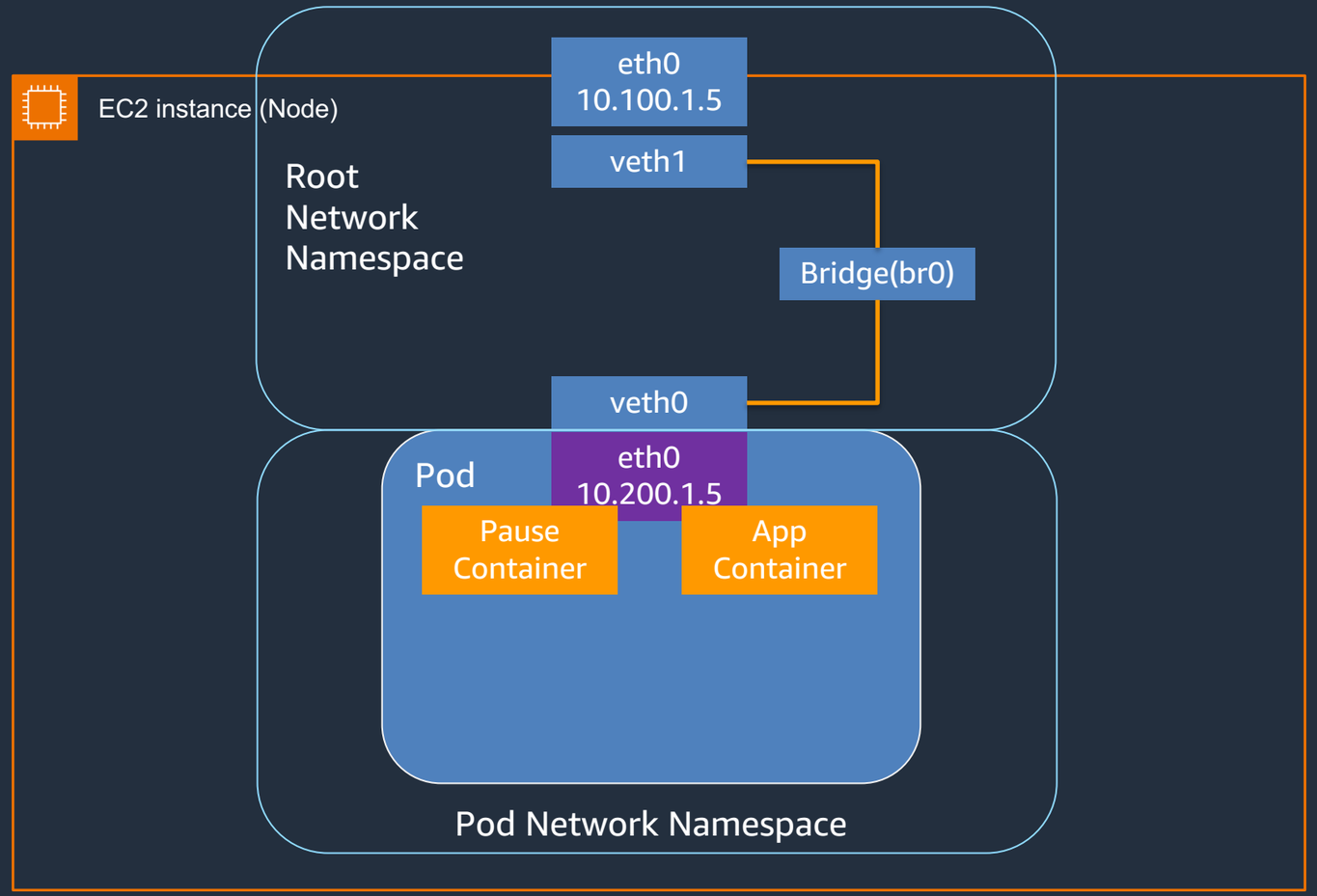
aws

Linux Networking Primitives for Kubernetes



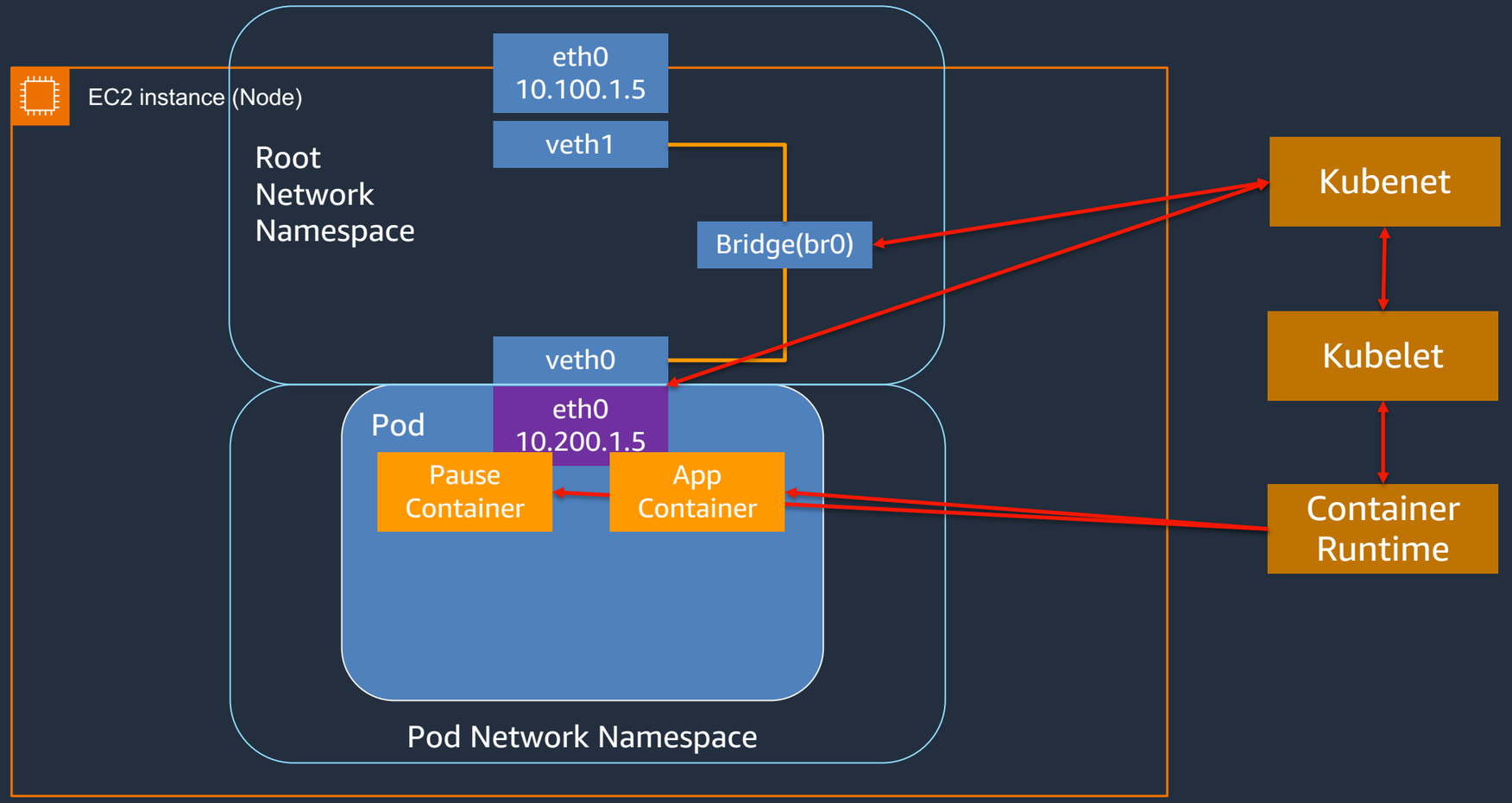
aws

Linux Networking Primitives for Kubernetes



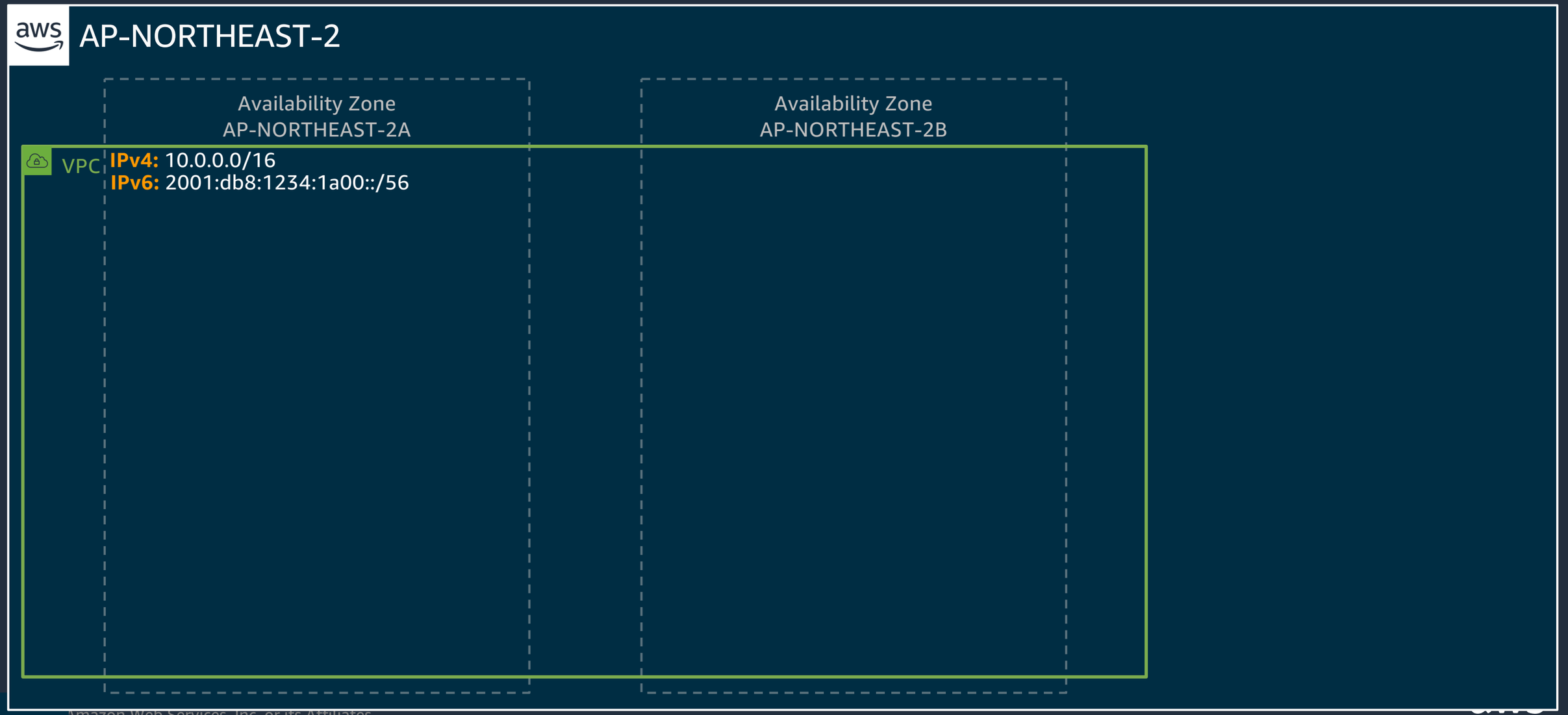
aws

Linux Networking Primitives for Kubernetes

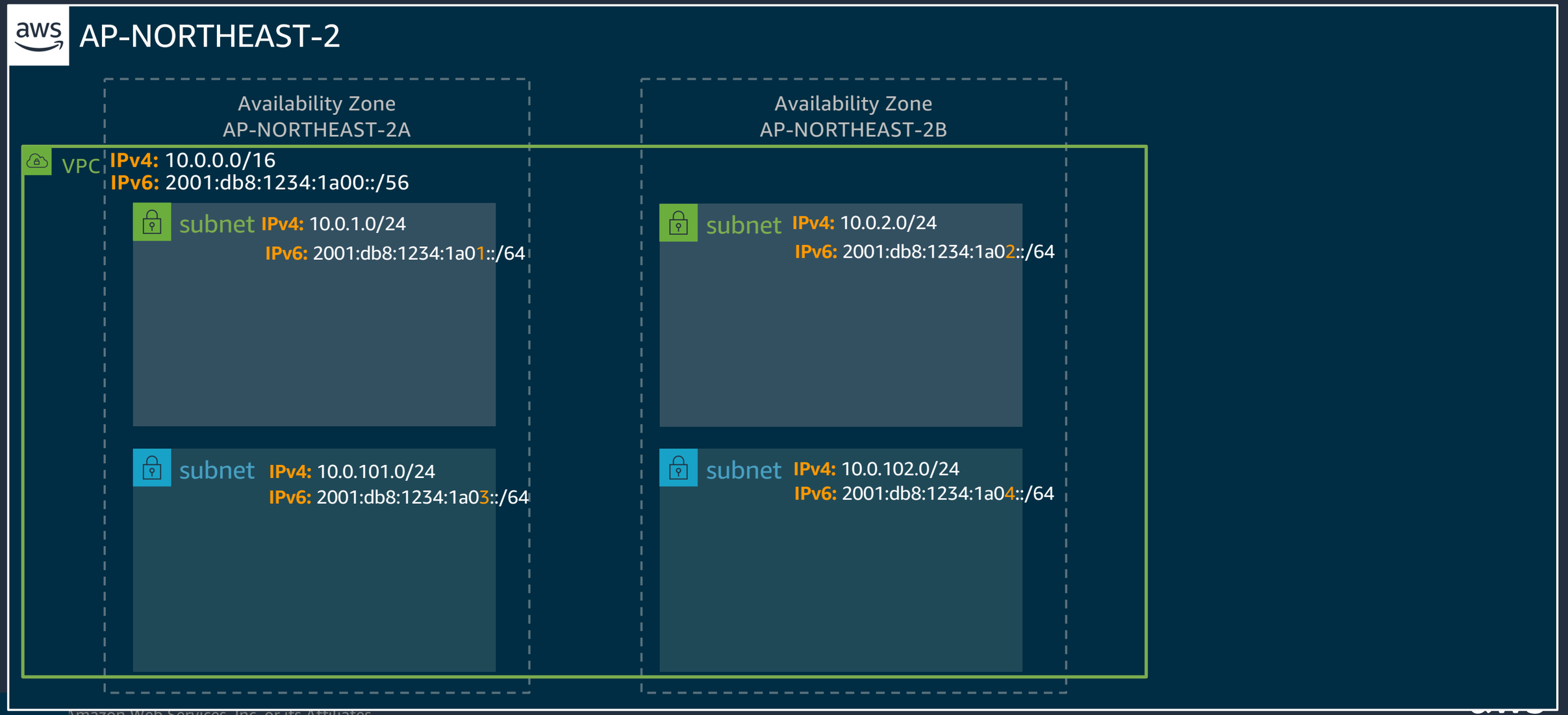


aws

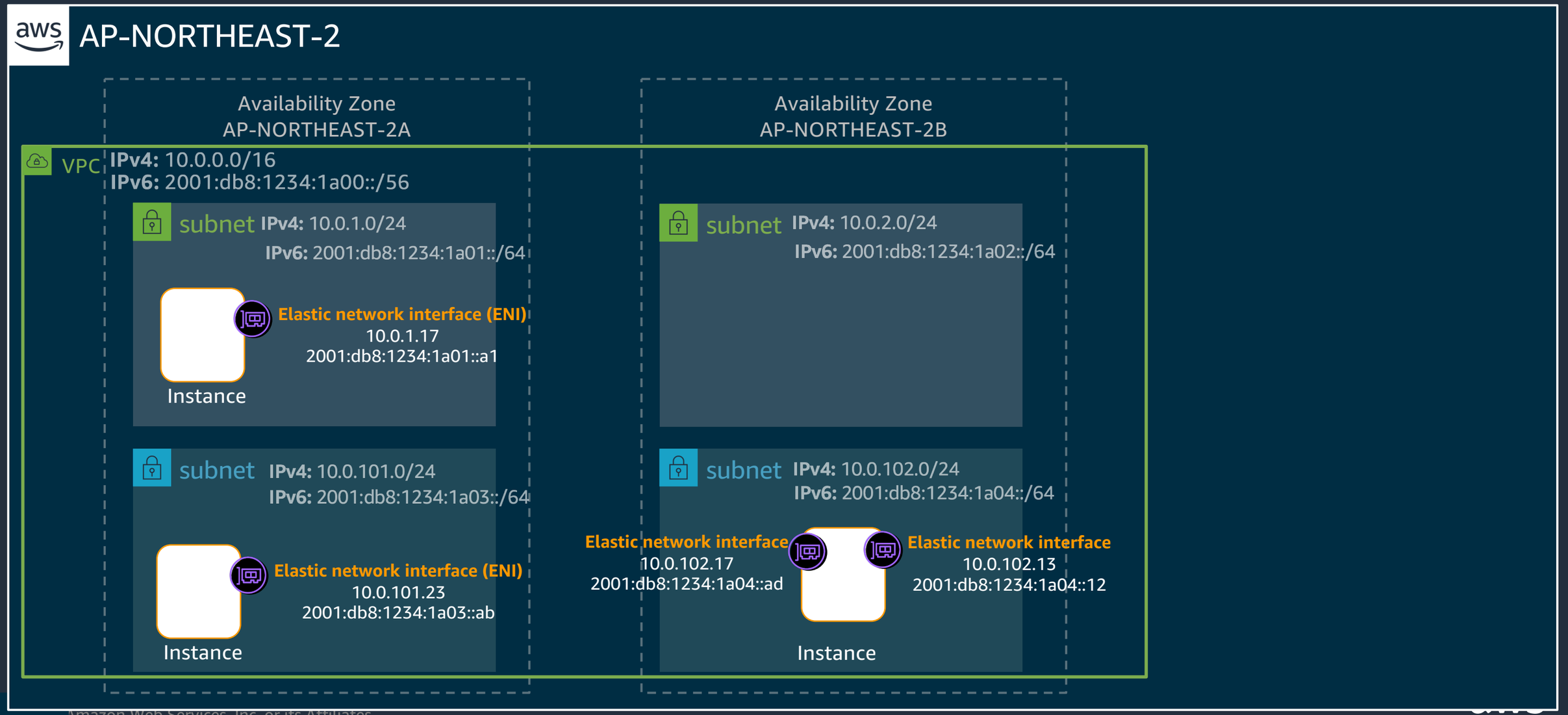
Amazon VPC – IP range (CIDR)



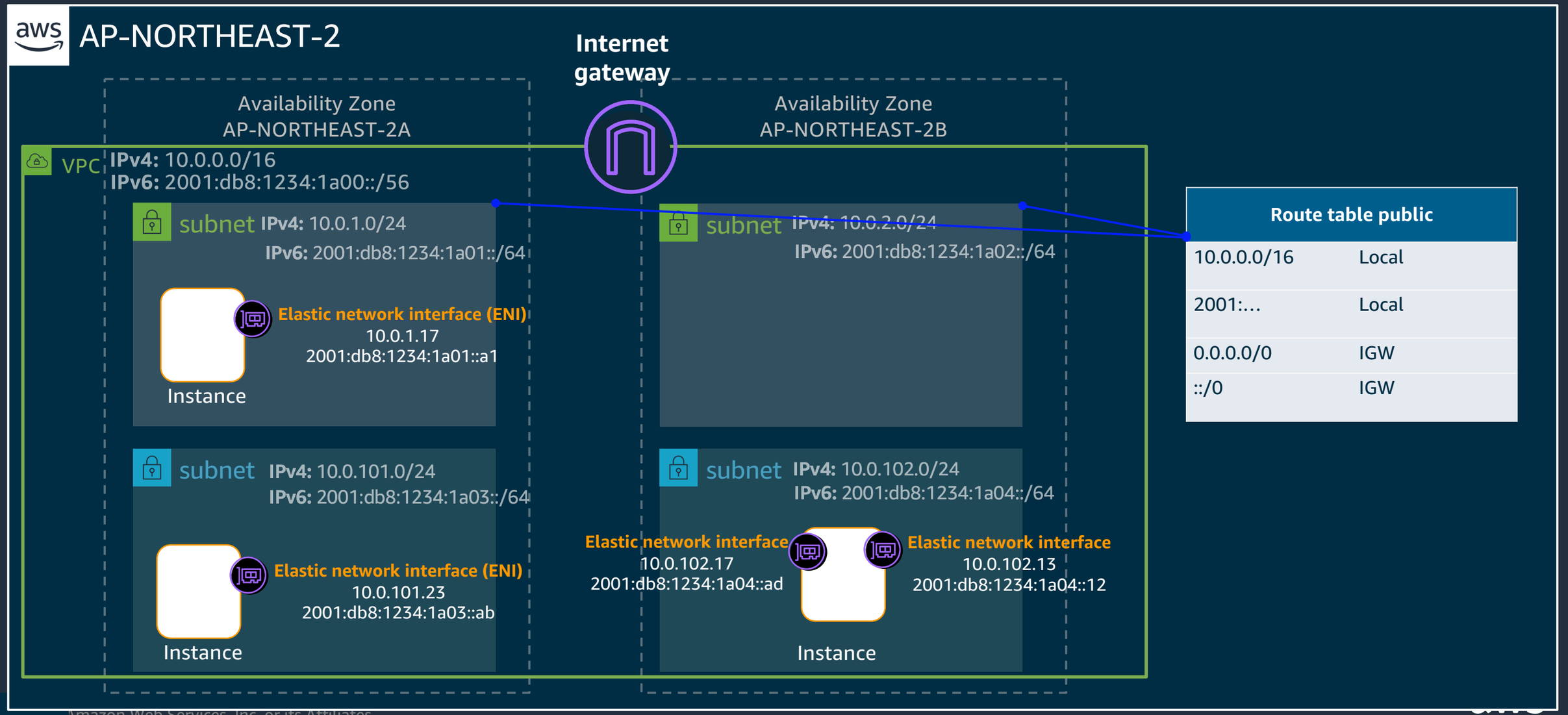
Amazon VPC – Subnets



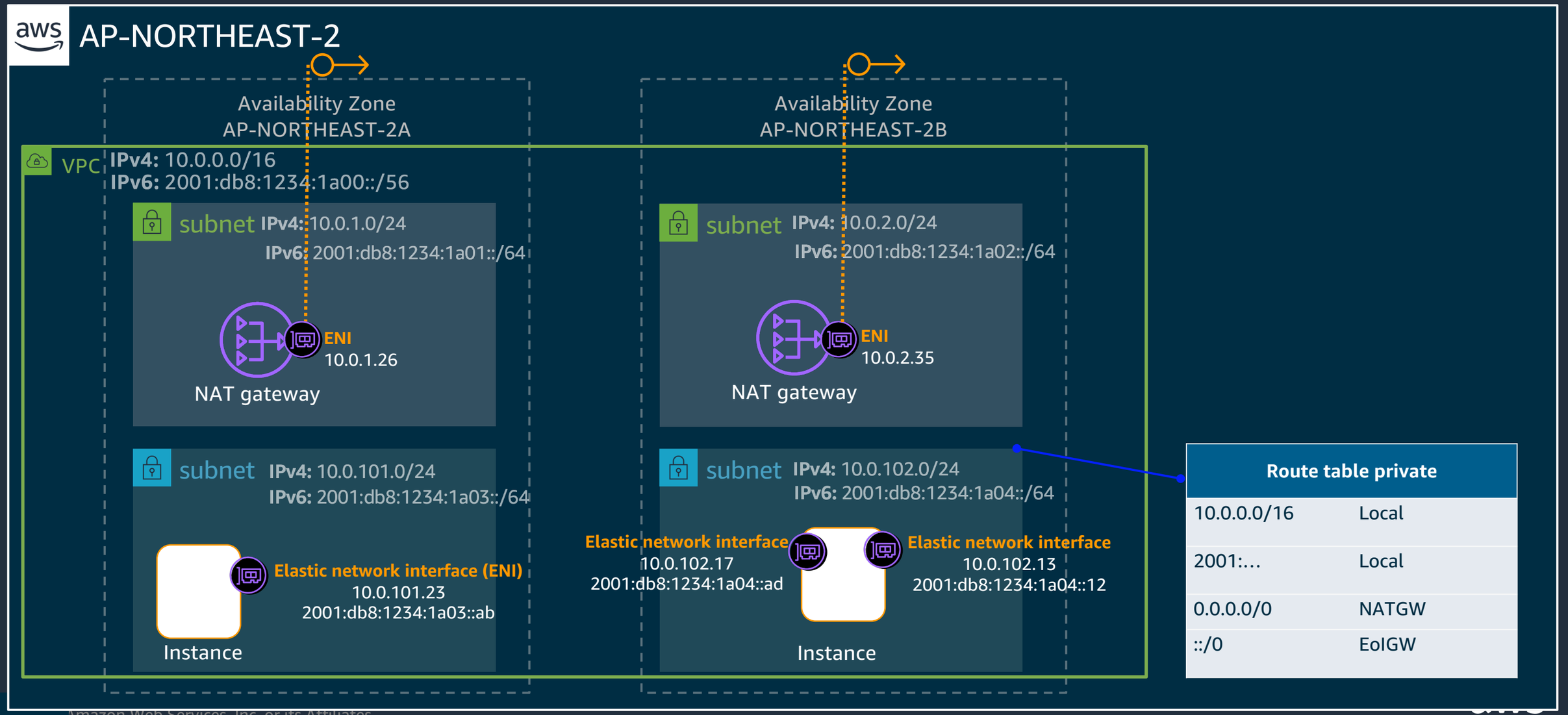
Amazon VPC – Interfaces



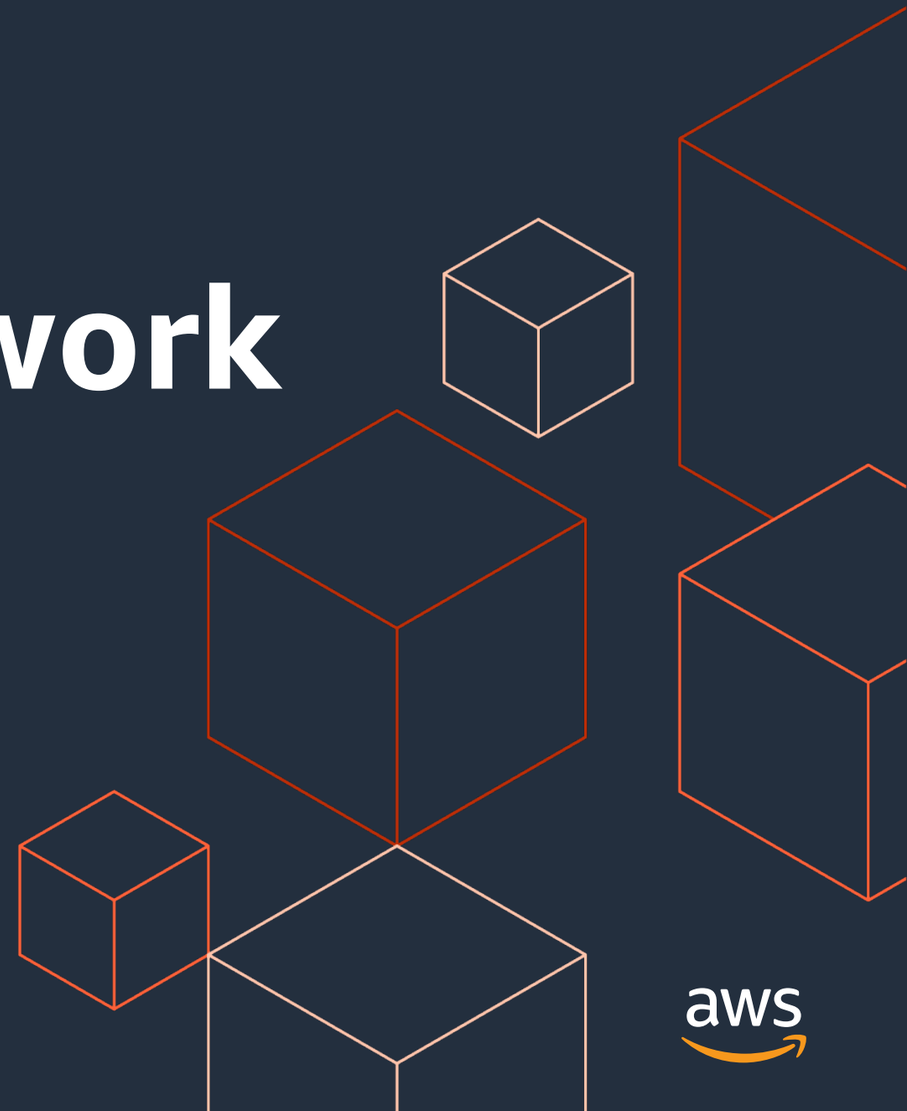
Amazon VPC – Connectivity to Internet



Amazon VPC – Connectivity to Internet

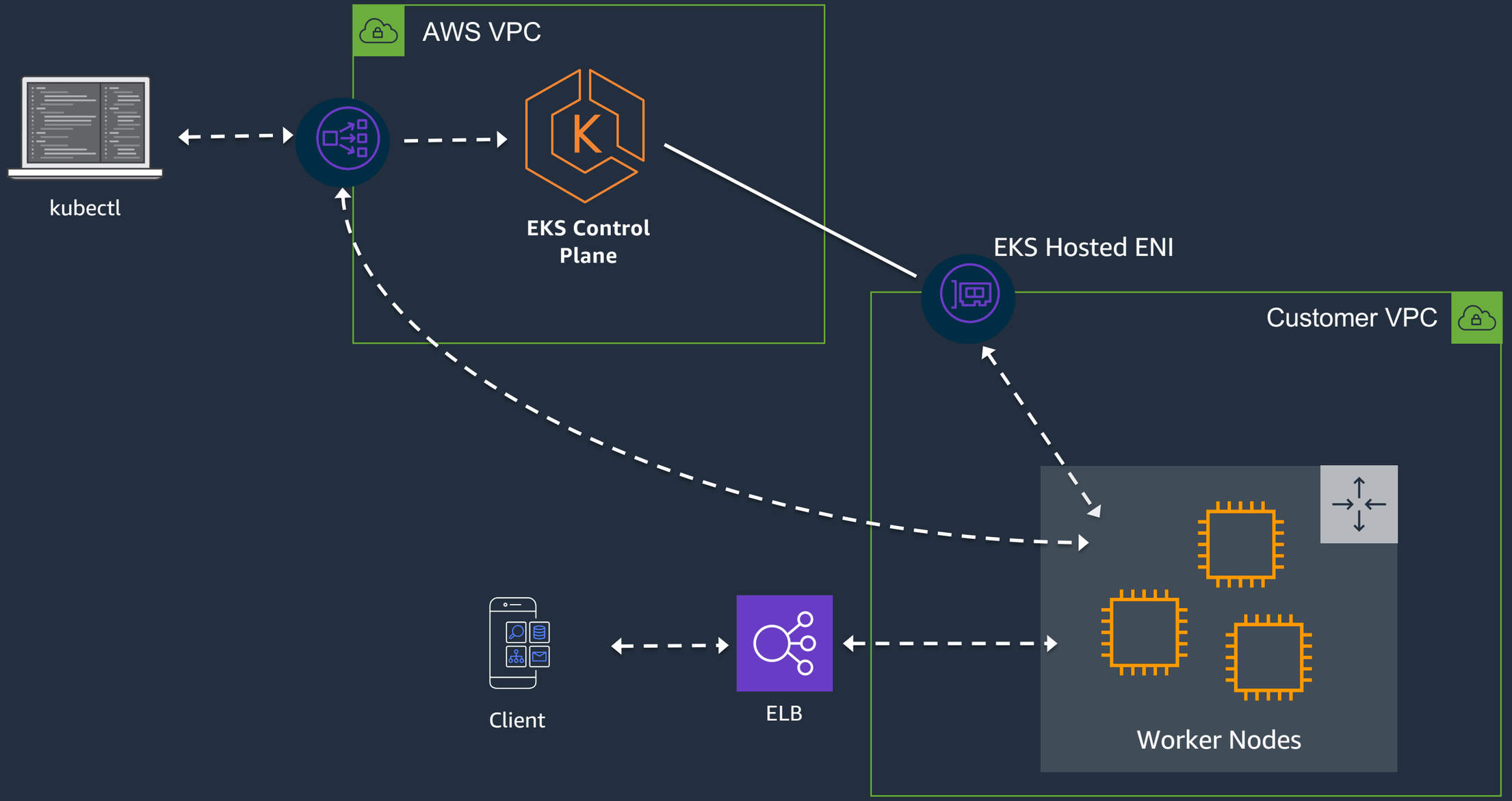


EKS Control Plane Network

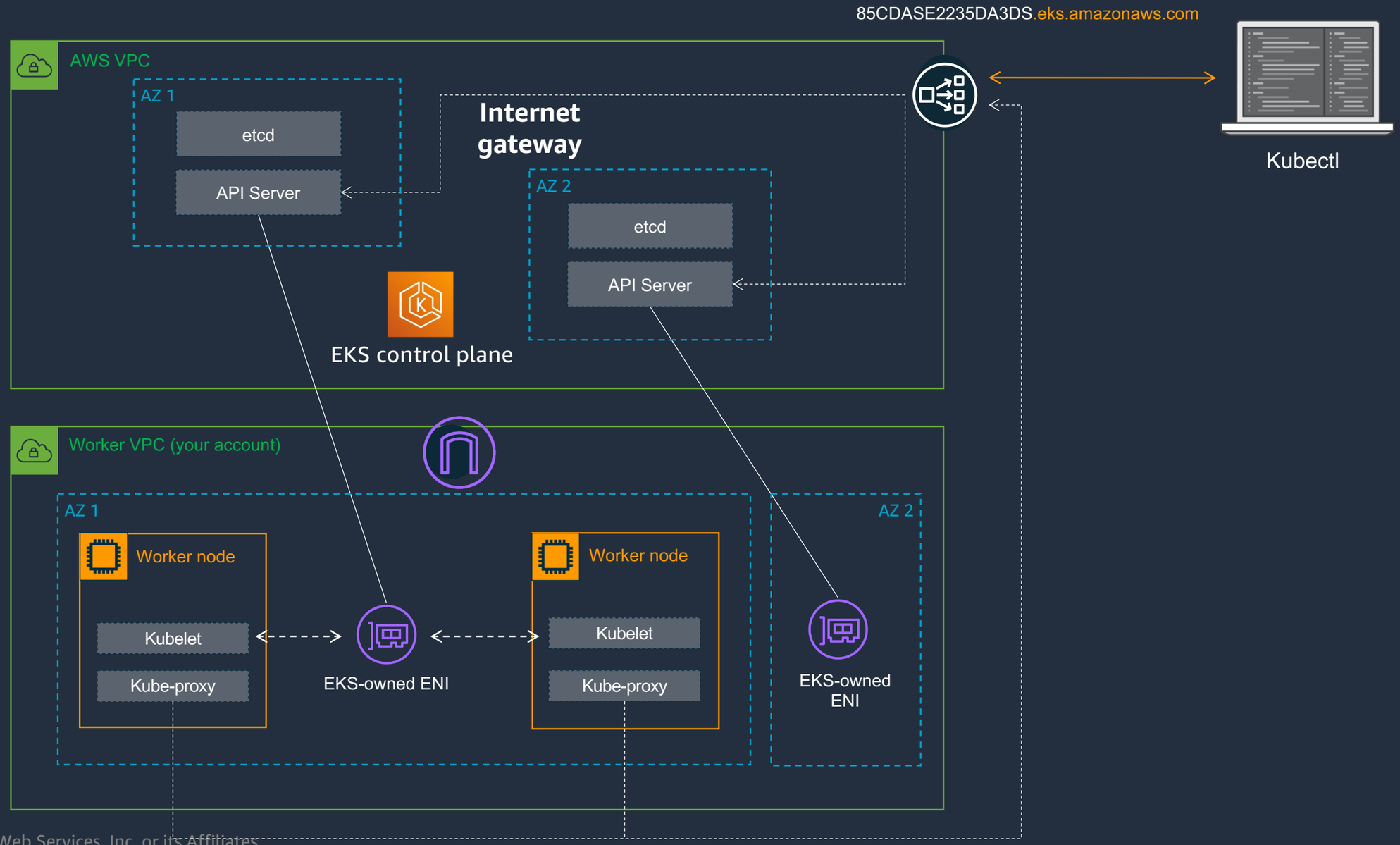


aws

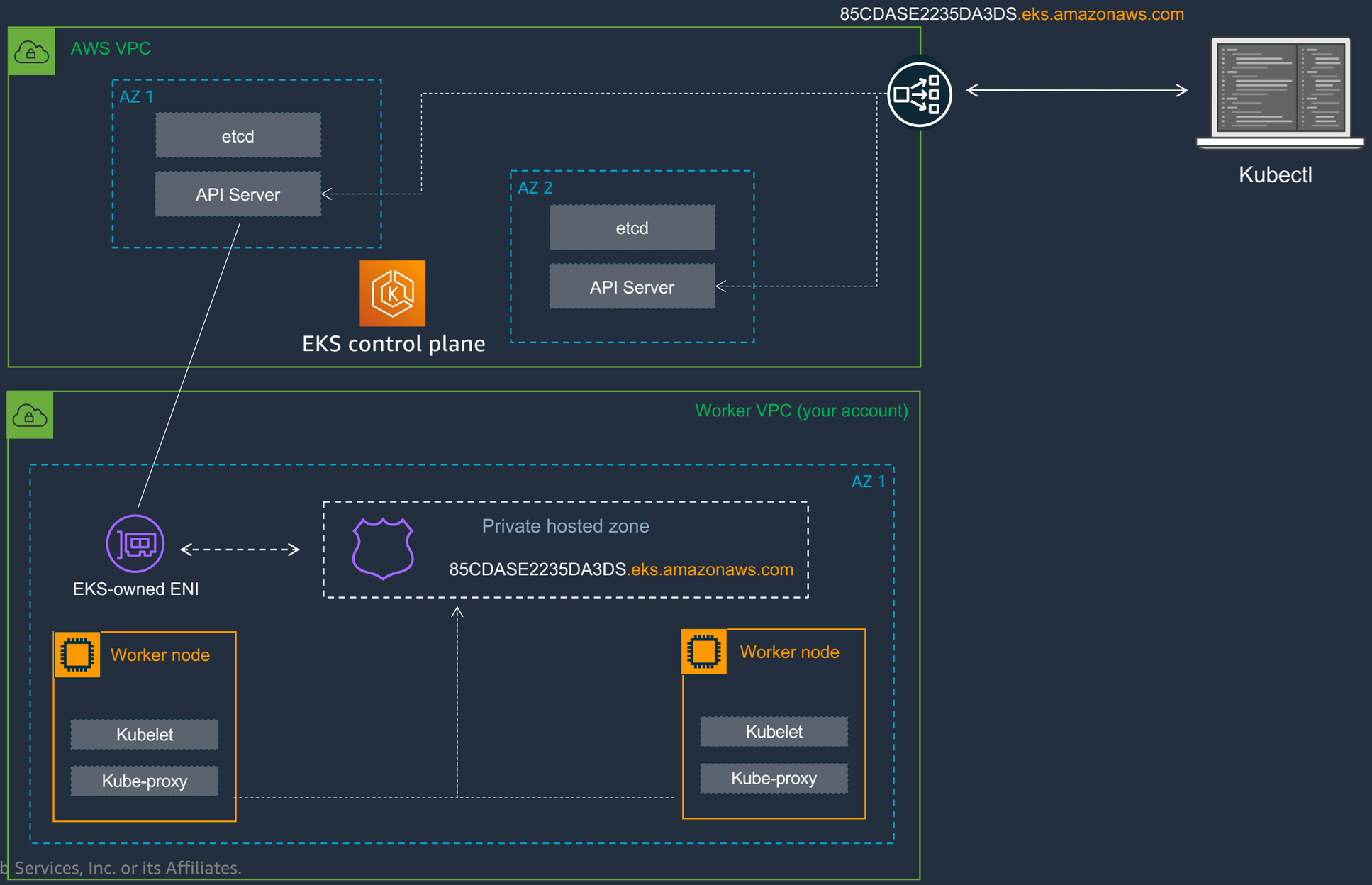
EKS Network Architecture



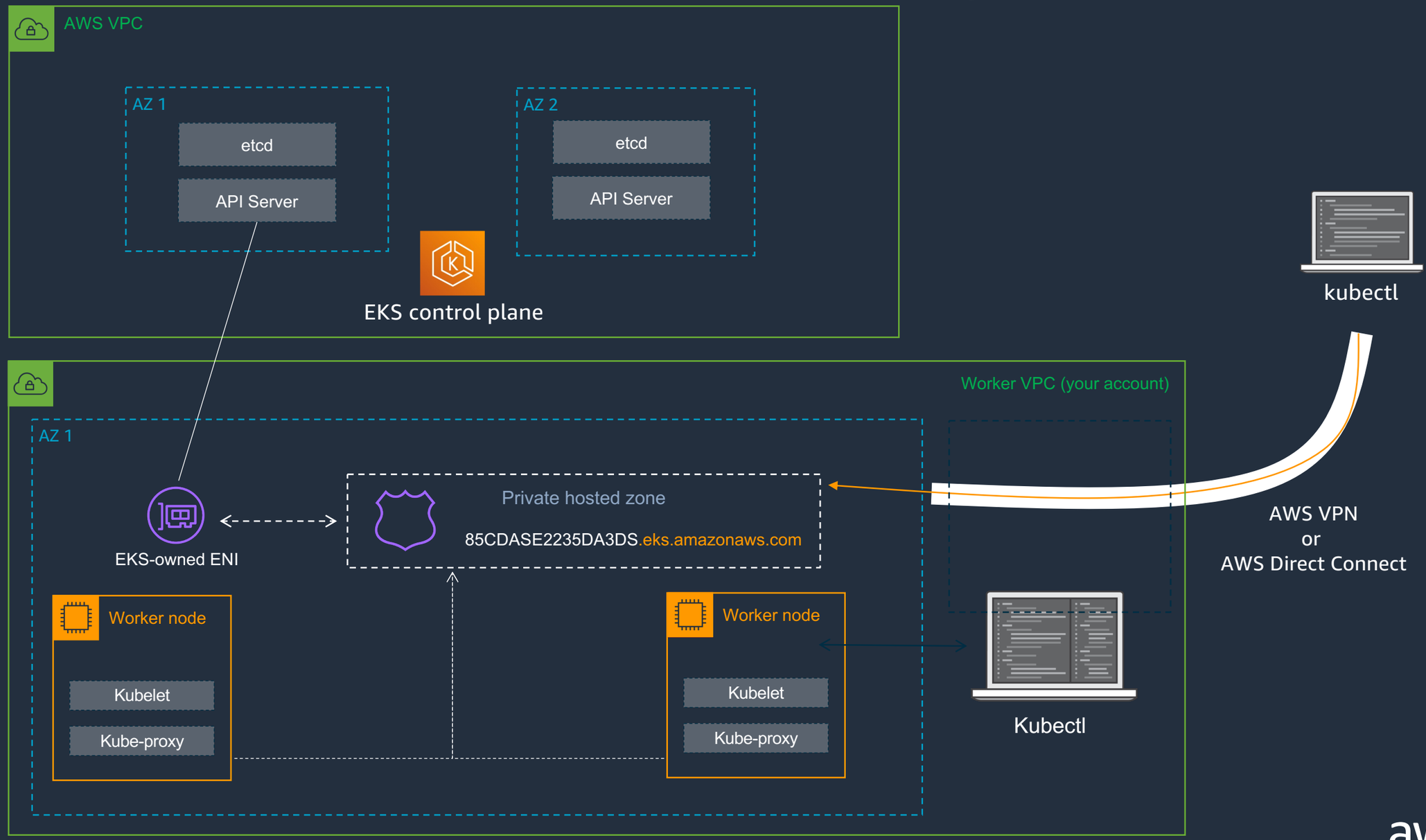
EKS cluster endpoint access – Public



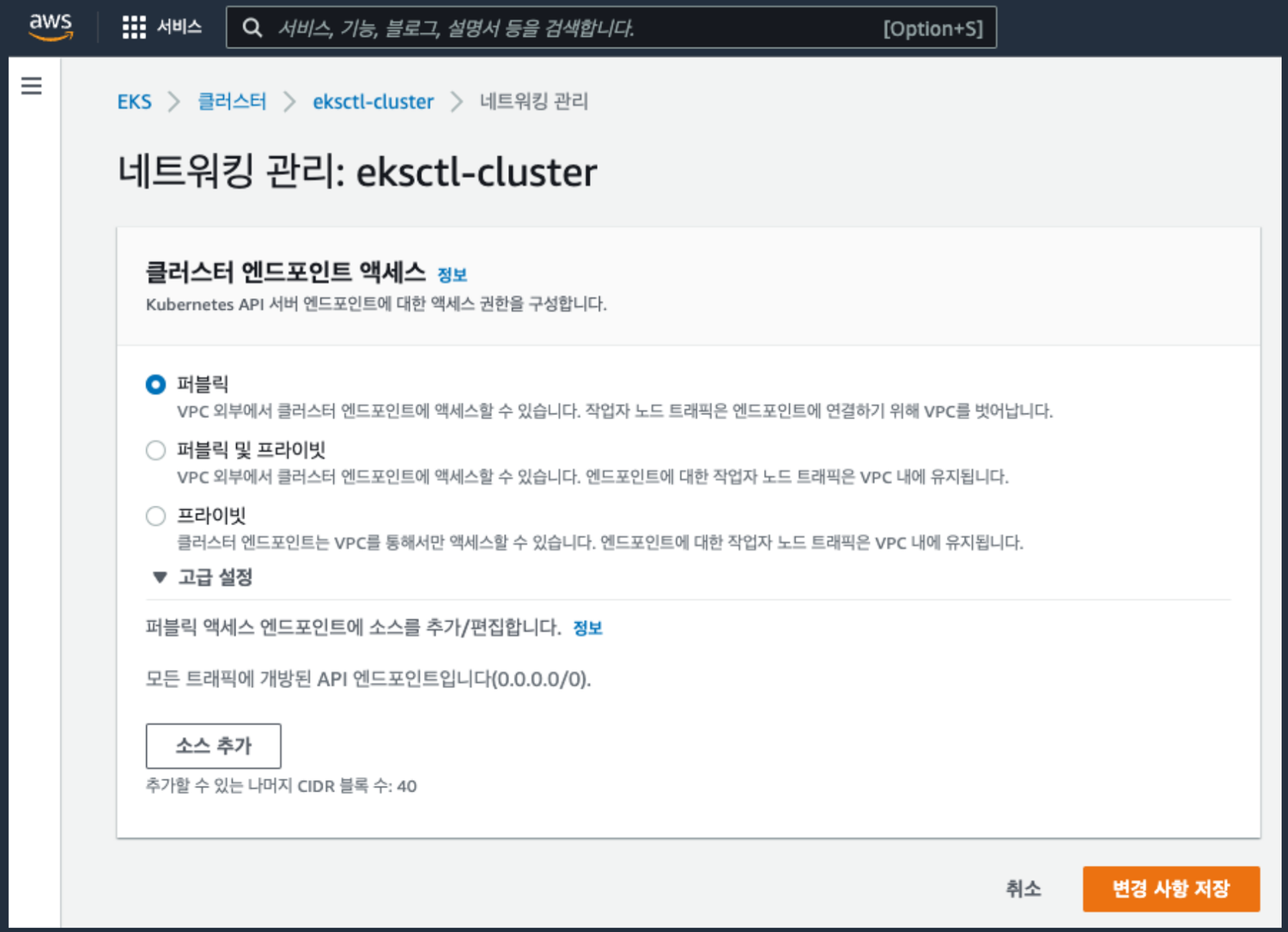
EKS cluster endpoint access – Public & private



EKS cluster endpoint access – Private only



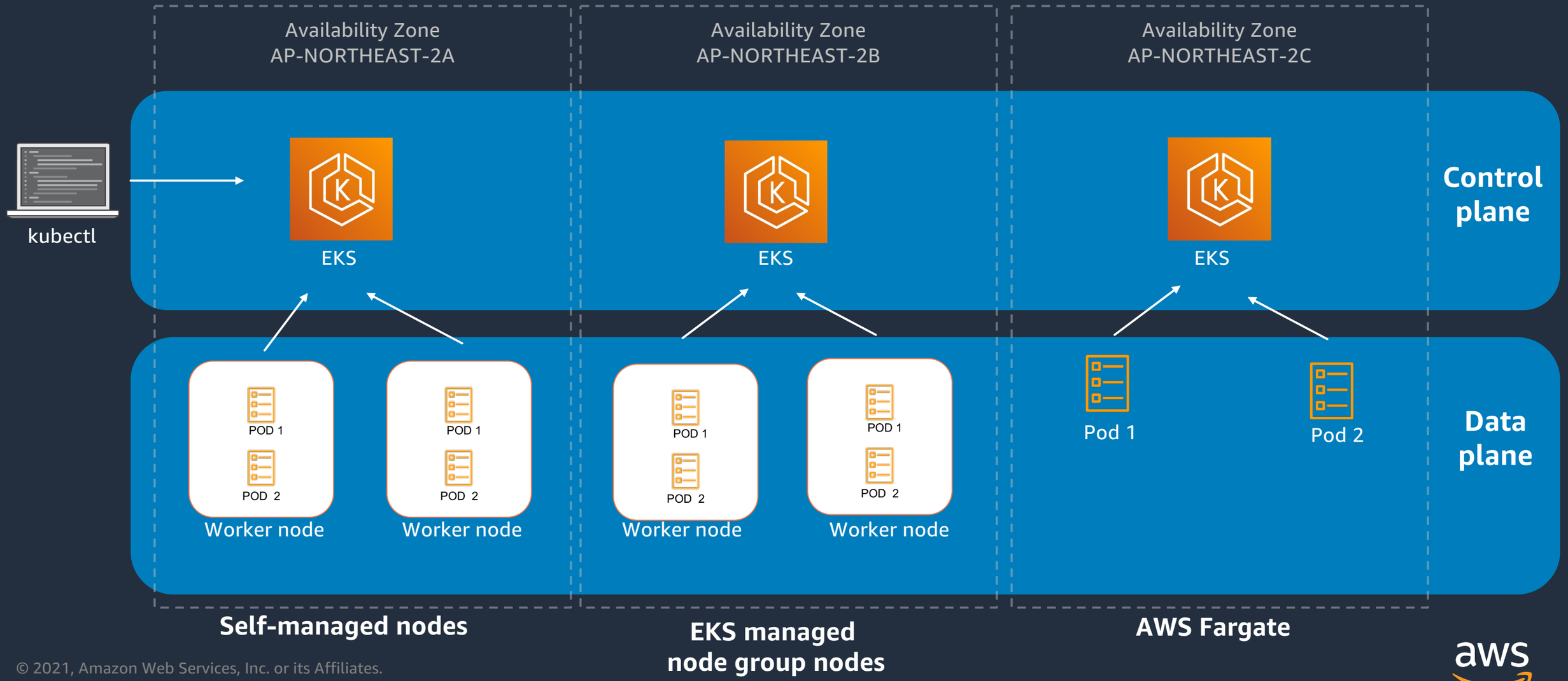
EKS cluster endpoint access



EKS Data Plane Network (CNI)

aws

Amazon EKS



Kubernetes networking model

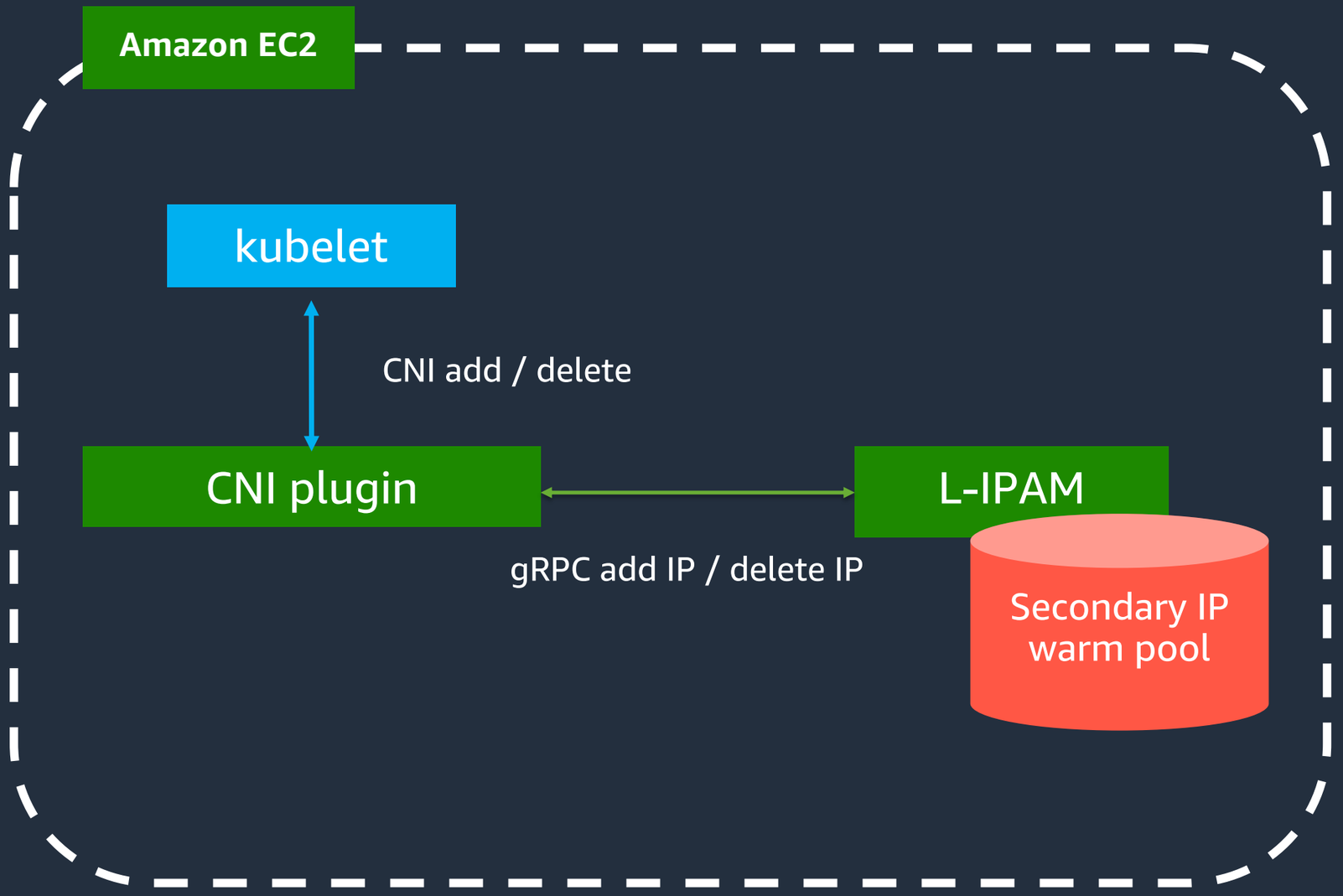
- Network Interface Specification for Kubernetes
- Assign a network namespace and a network interface to the Pod at startup
- Every Pod gets its own IP address
- Containers in the same Pod share the network (IP address)
- Pods communicate to other Pods without NAT
- Clean up the network namespace when the Pod terminates, frees up the IP address

CNI

Container Network Interface plugin

aws

Amazon VPC CNI

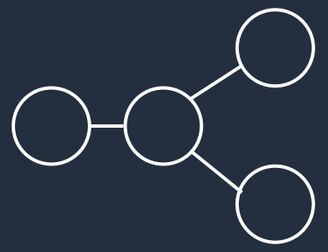


Customizable

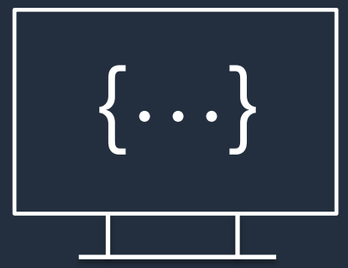
- WARM_ENI_TARGET
- WARM_IP_TARGET
- MINIMUM_IP_TARGET
- WARM_PREFIX_TARGET

aws

Amazon VPC CNI plugin



Native VPC networking with CNI plugin



Pods have the same VPC address inside the pod as on the VPC

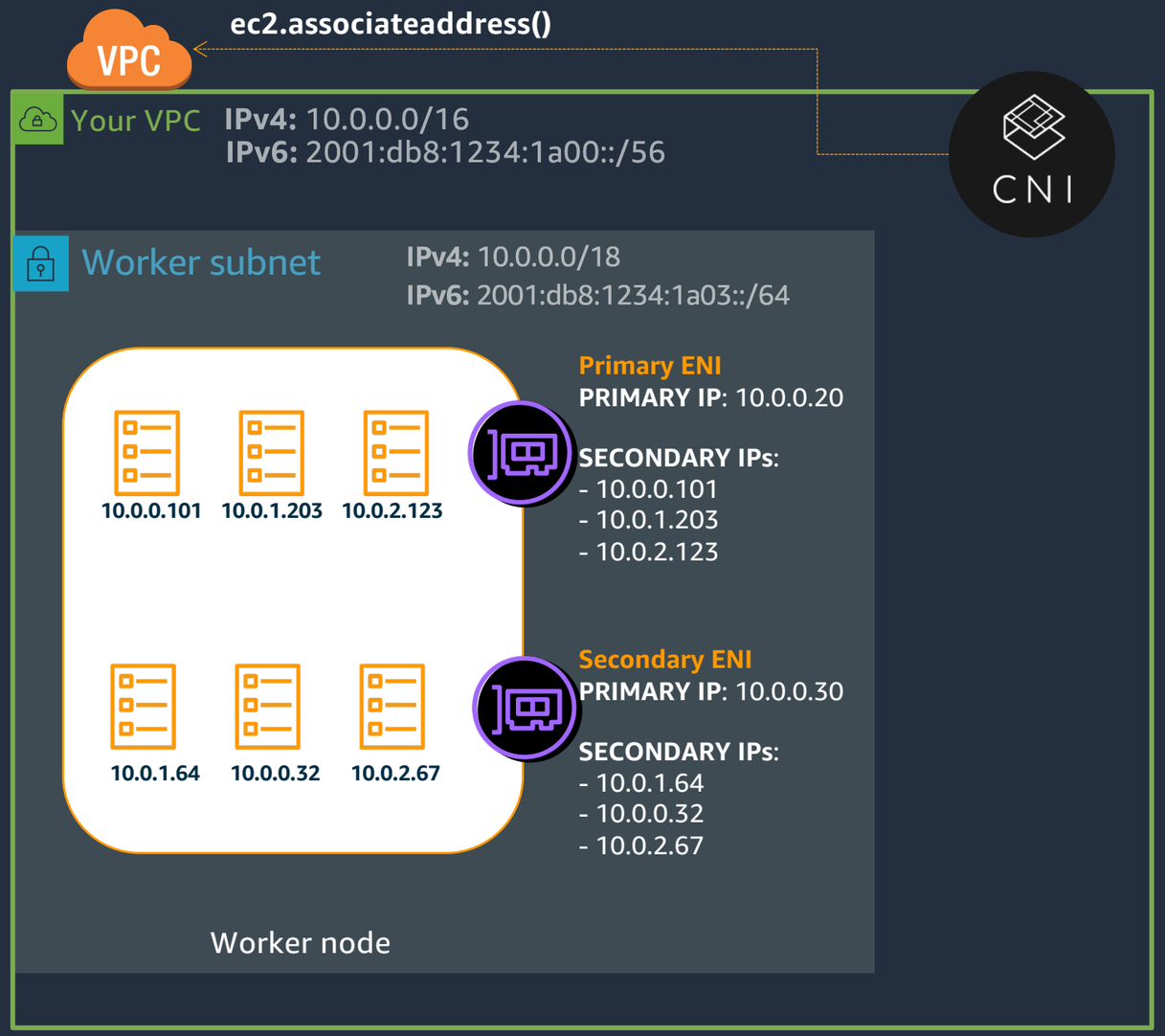
Simple, secure networking

Open source and on Github

https://github.com/aws/amazon-vpc-cni-k8s

aws

Pod IPv4 networking – CNI plugin



The Amazon VPC Container Network Interface (CNI) plugin is used for

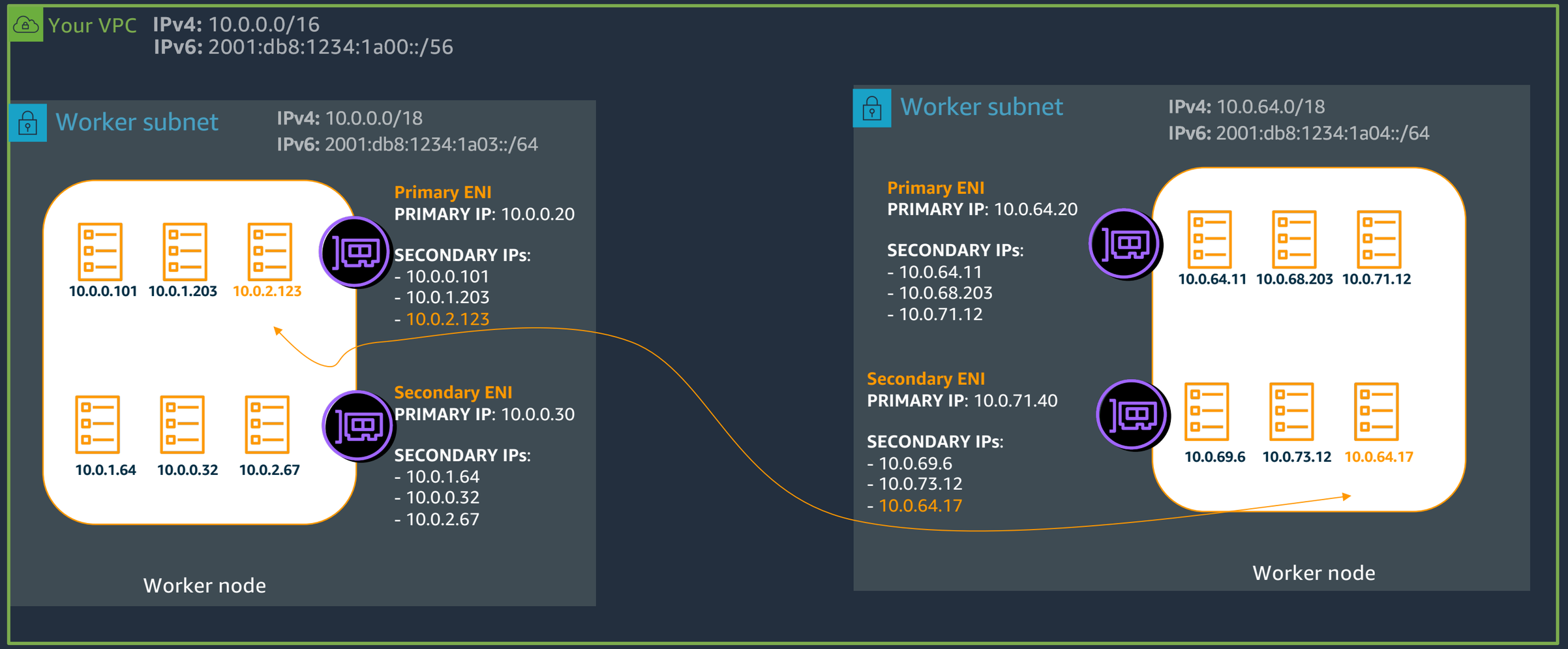
- Creating and attaching ENIs to worker nodes
- Assigning secondary IP addresses for Pods
- Wiring the host network

Alternatives to Amazon VPC CNI

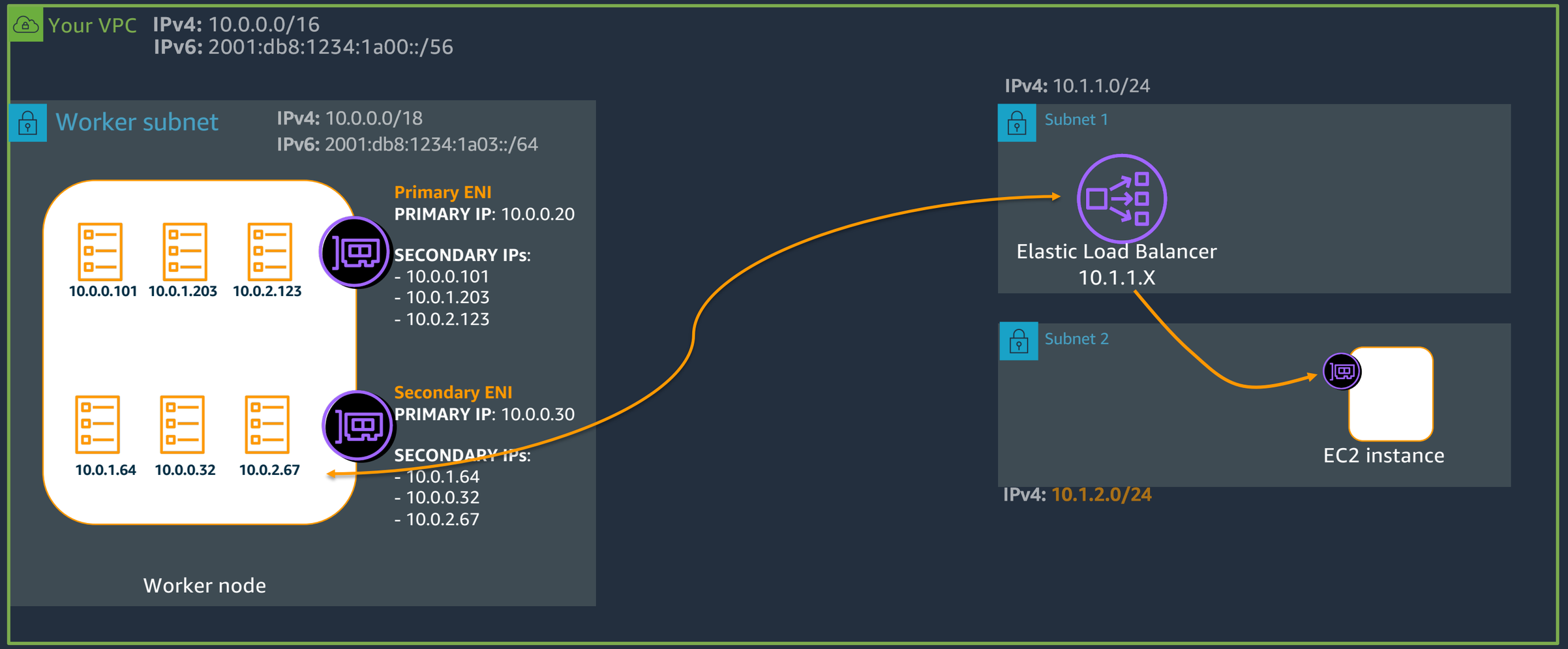
- Calico from Tigera
- Cilium from Isovalent
- Weave Net from Weaveworks
- Antrea from VMware

aws

Pod IPv4 networking – Pod to Pod



Pod IPv4 networking – Pod to Pod



Amazon CNI – Configuration options

- Source NAT

```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_EXTERNALSNAT=true/false
```

- Prefix delegation

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true/false
```

- Security groups for Pods

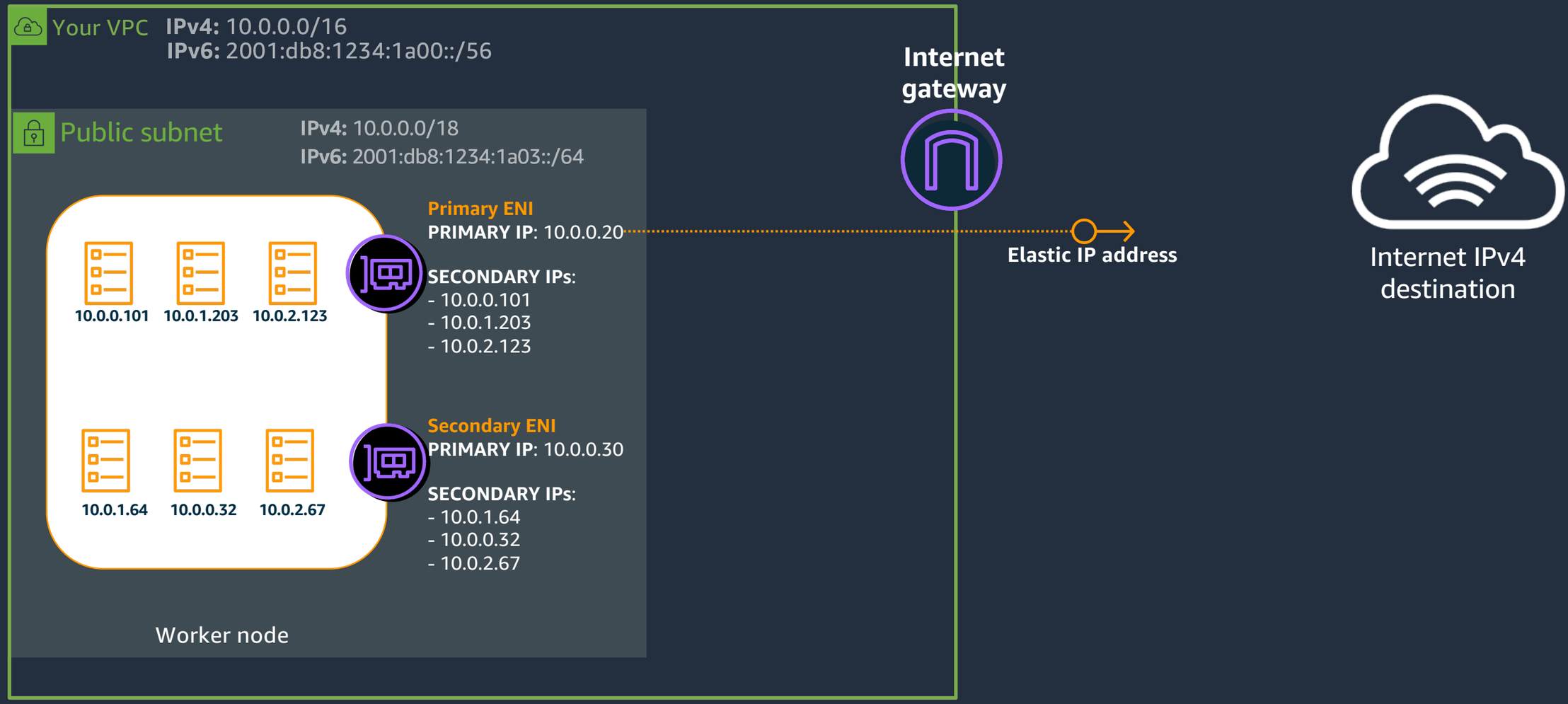
```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true/false
```

- Custom networking

```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true/false
```

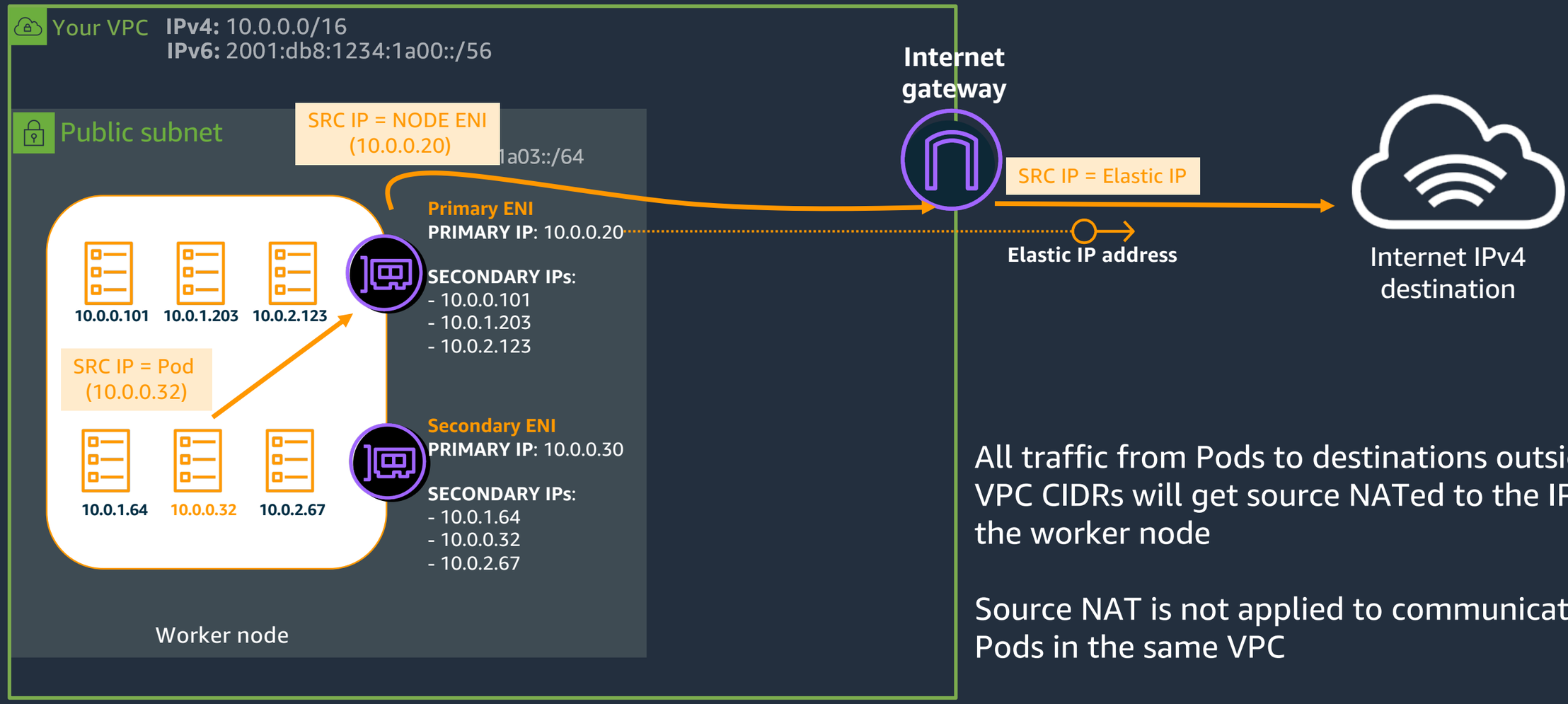

aws

Pod IPv4 networking – Source NAT enabled




```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_EXTERNALSNAT=false
```

Pod IPv4 networking – Source NAT enabled

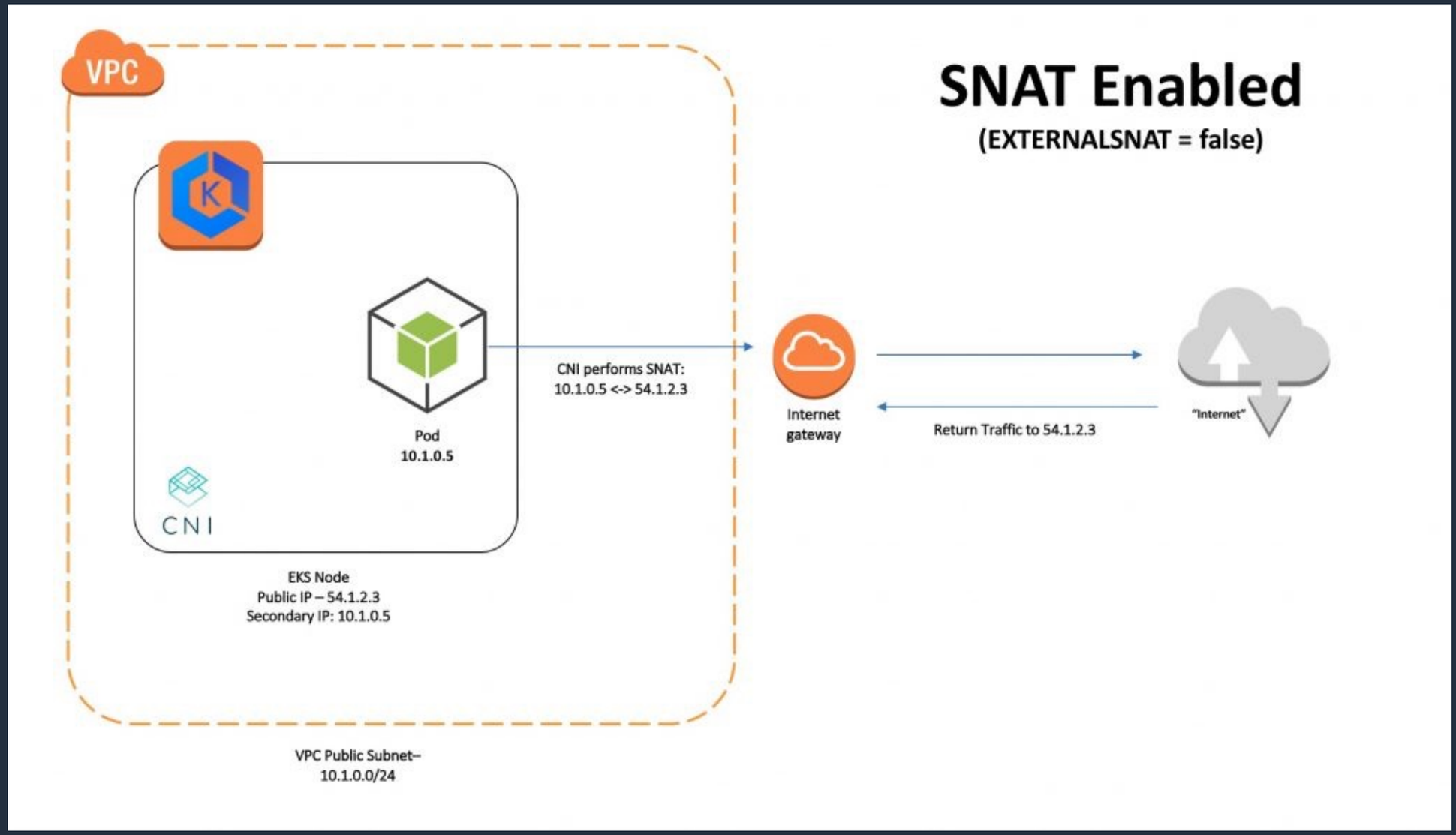


All traffic from Pods to destinations outside of the VPC CIDRs will get source NATed to the IP address of the worker node

Source NAT is not applied to communication between Pods in the same VPC

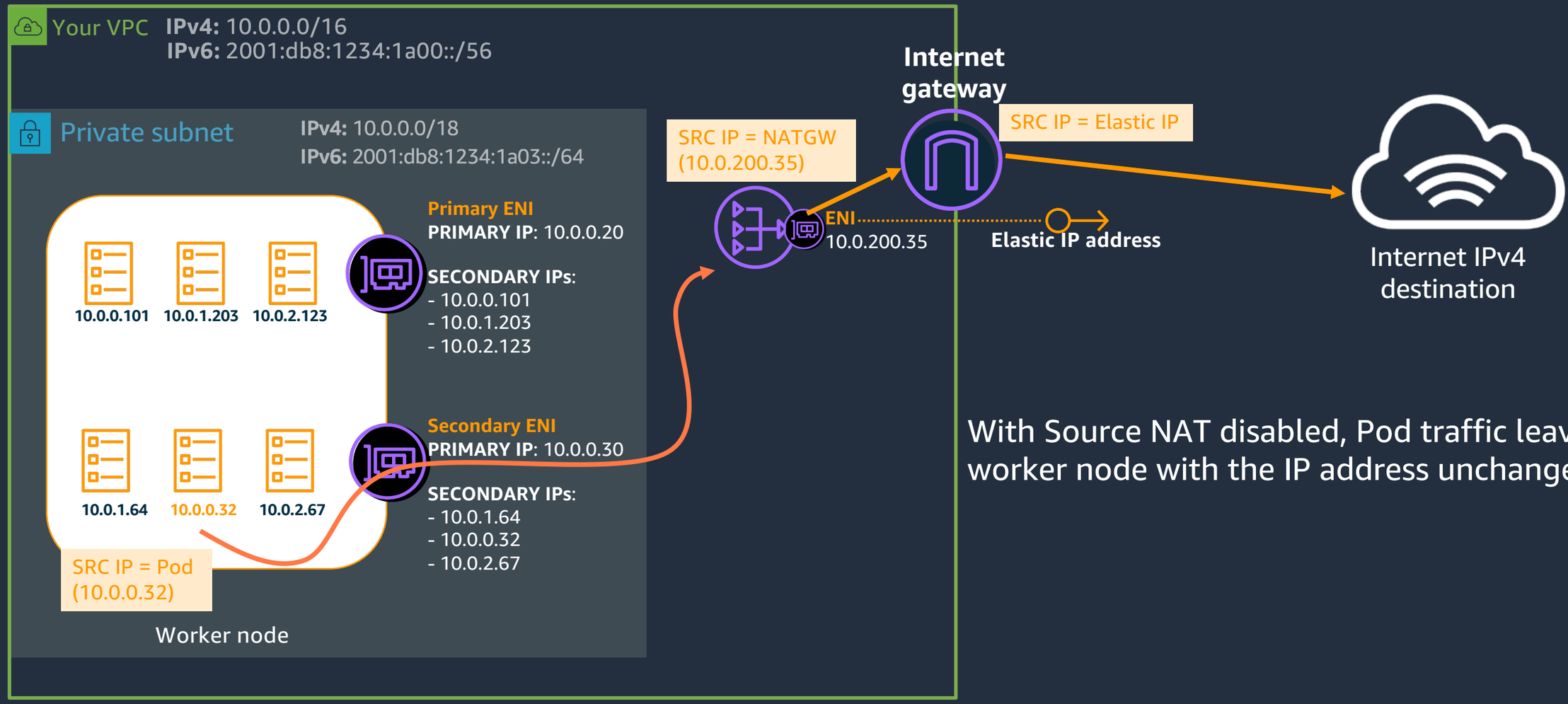
```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_EXTERNALSNAT=false
```

Pod IPv4 networking – Source NAT enabled



aws

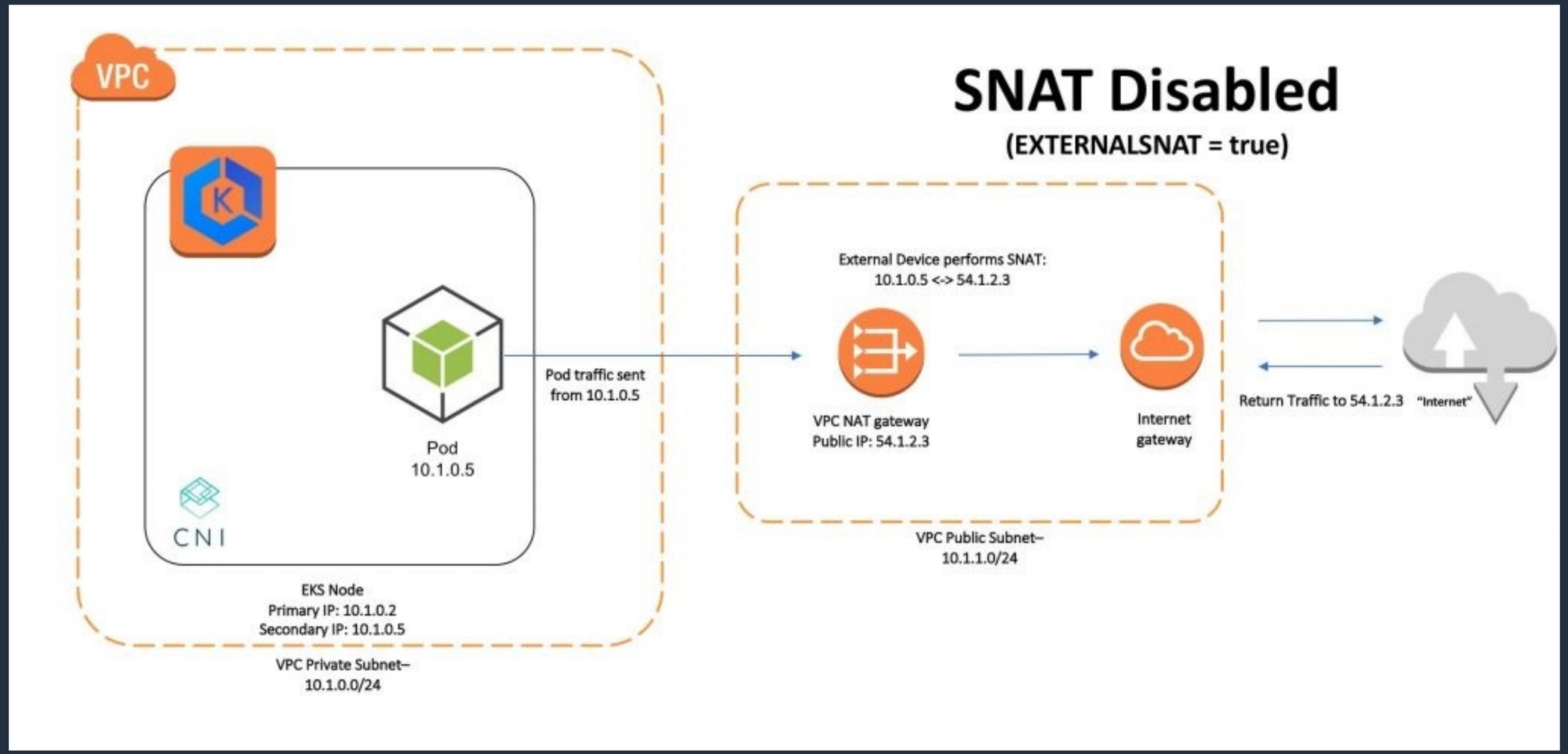
Pod networking – Source NAT disabled



With Source NAT disabled, Pod traffic leaves the worker node with the IP address unchanged

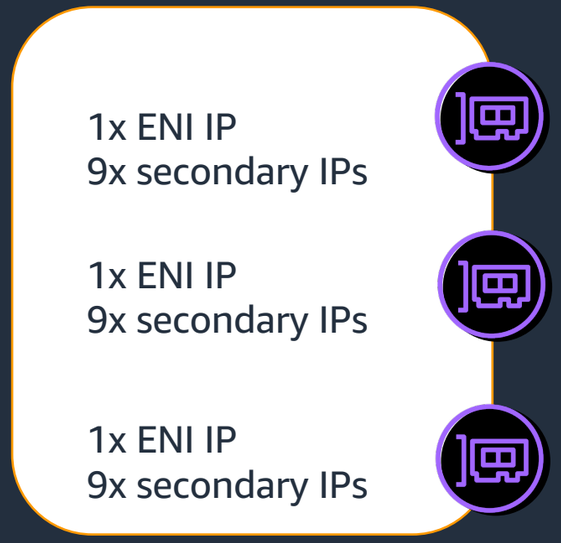
```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

Pod networking – Source NAT disabled



aws

Secondary IPs for Pods per EC2 instance



m5.large

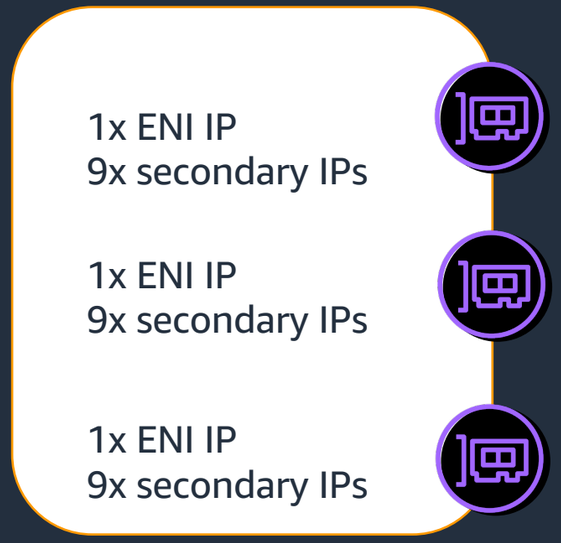
Instance type	Maximum network interfaces	Private IPv4 addresses per interface
m5.large	3	10

MAX PODS = (Number of network interfaces × [the number of IP addresses per network interface – 1]) + 2

?

aws

Secondary IPs for Pods per EC2 instance



m5.large

Instance type	Maximum network interfaces	Private IPv4 addresses per interface
m5.large	3	10

https://github.com/awslabs/amazon-eks-ami/blob/master/files/eni-max-pods.txt

MAX PODS = (Number of network interfaces × [the number of IP addresses per network interface – 1]) + 2

29

aws

Amazon VPC prefix delegation

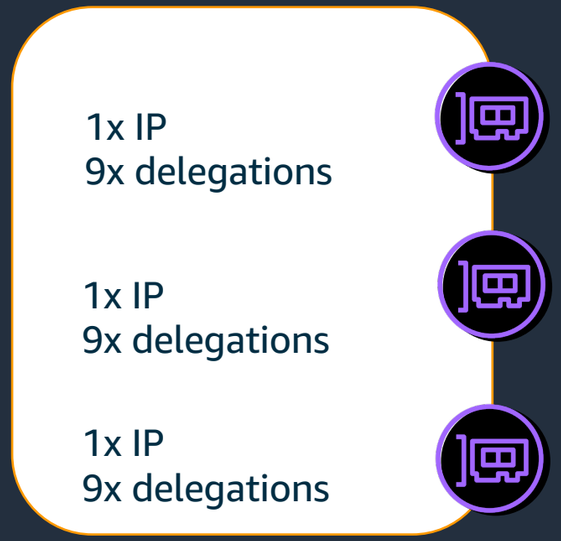
Allows for assigning a prefix to an EC2 ENI

- /28 block for IPv4 (16x IPv4 addresses)
- /80 block for IPv6 (280 trillion IPv6 addresses)

* Prefix delegation is only supported on Nitro instances

aws

Prefix delegation



m5.large

Instance type	Maximum network interfaces	Private IPv4 addresses per interface
m5.large	3	10

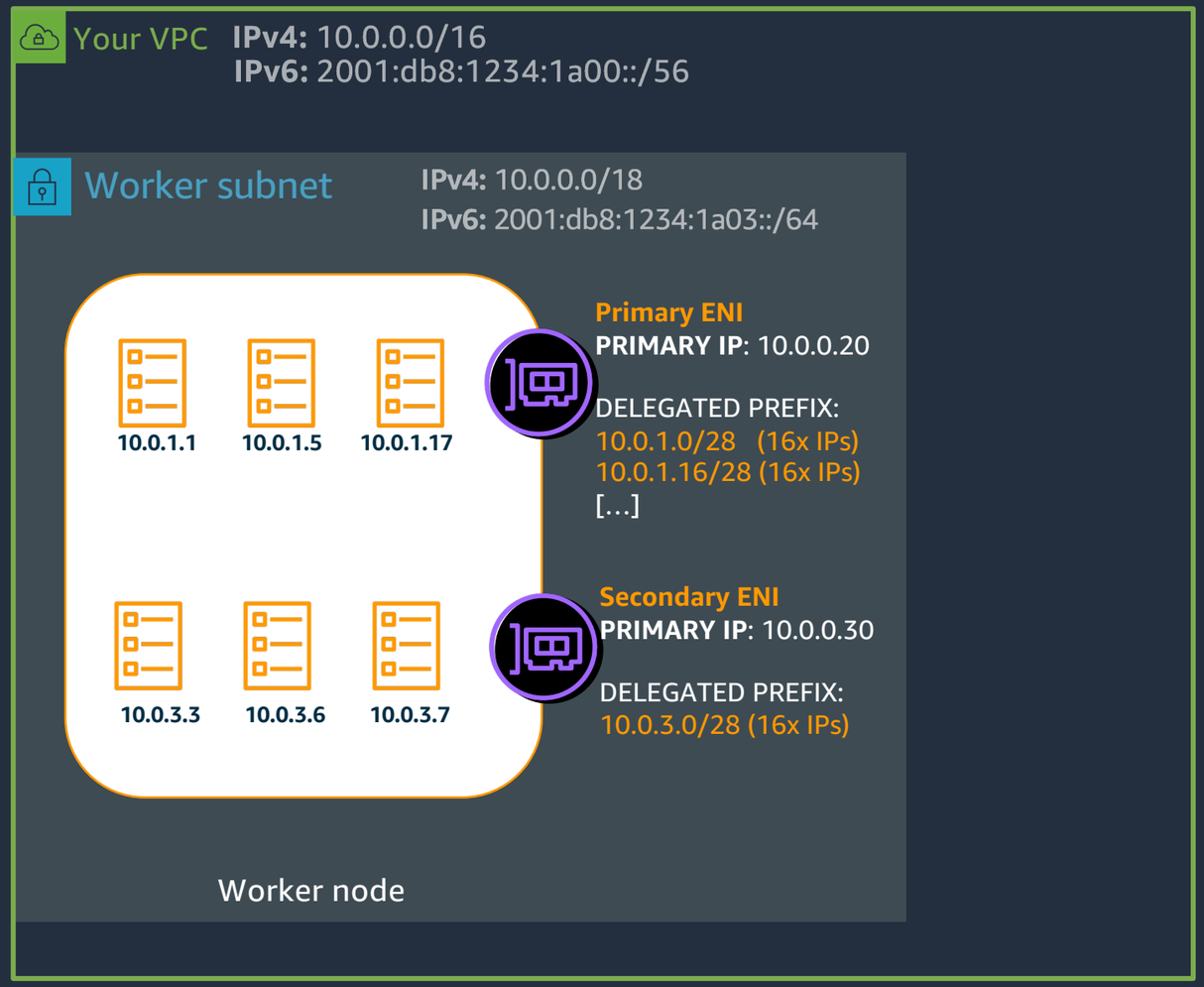
MAX IPs ≠ **MAX PODS**

432 IPv4
Trillions IPv6

110

aws

Pod IPv4 networking – Prefix delegation



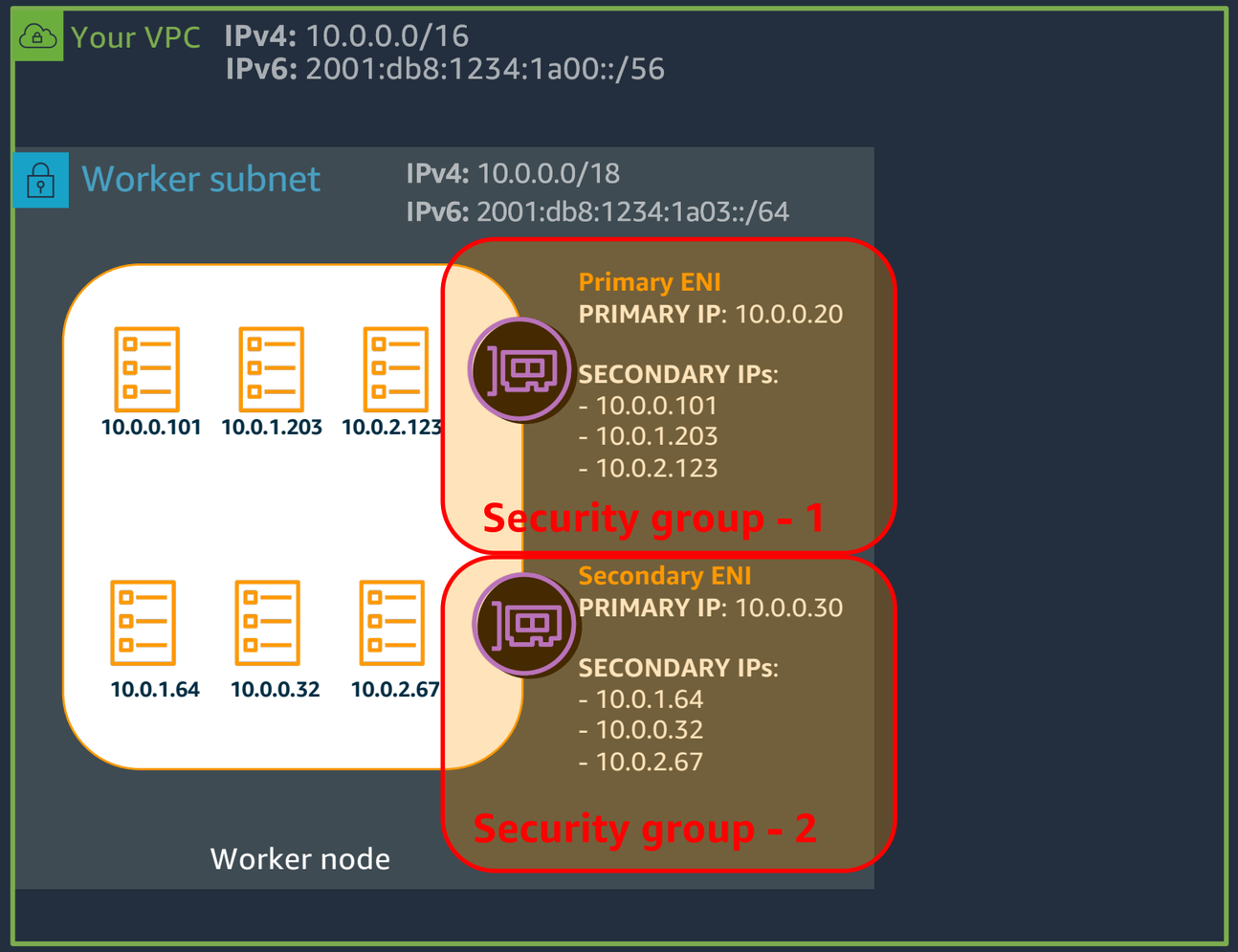
Benefits

- Increased Pod density
- Fewer API calls required to EC2 control plane

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
```


aws

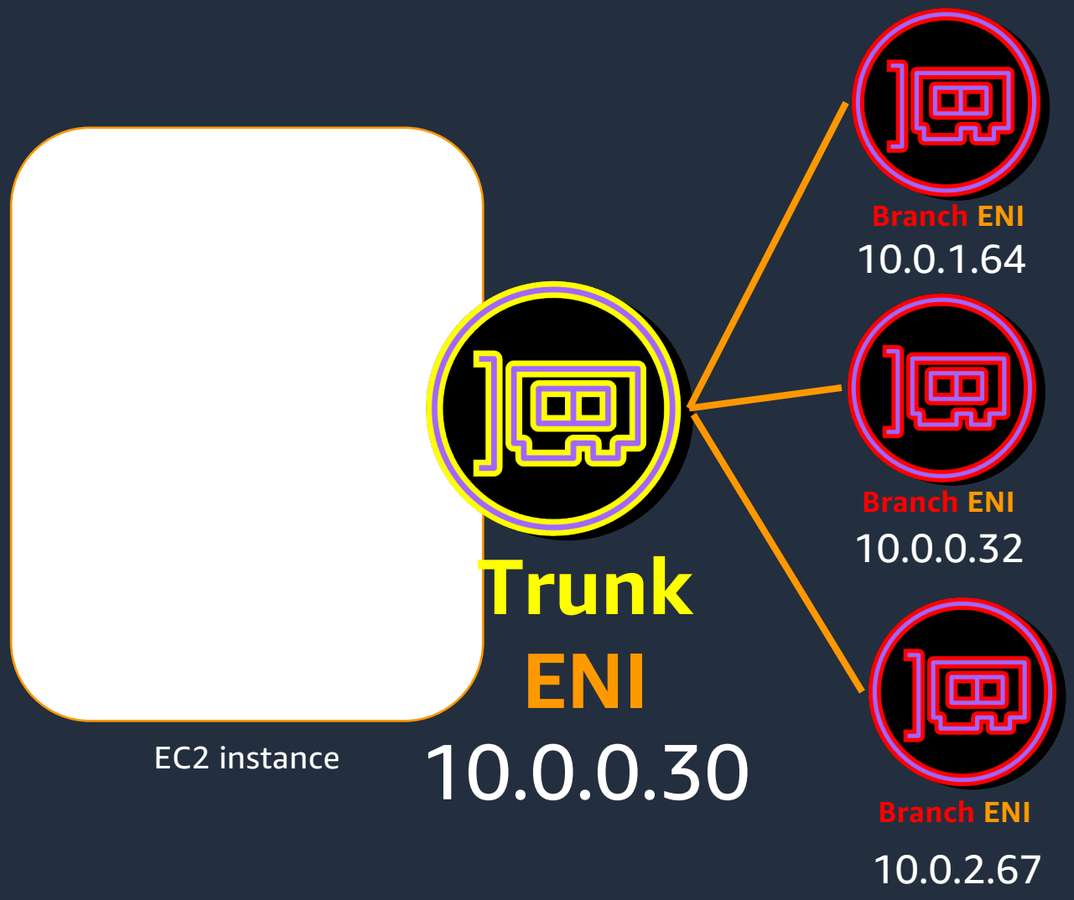
Pod networking – Security groups for Pods



Pods share the security group on their respective ENI

aws

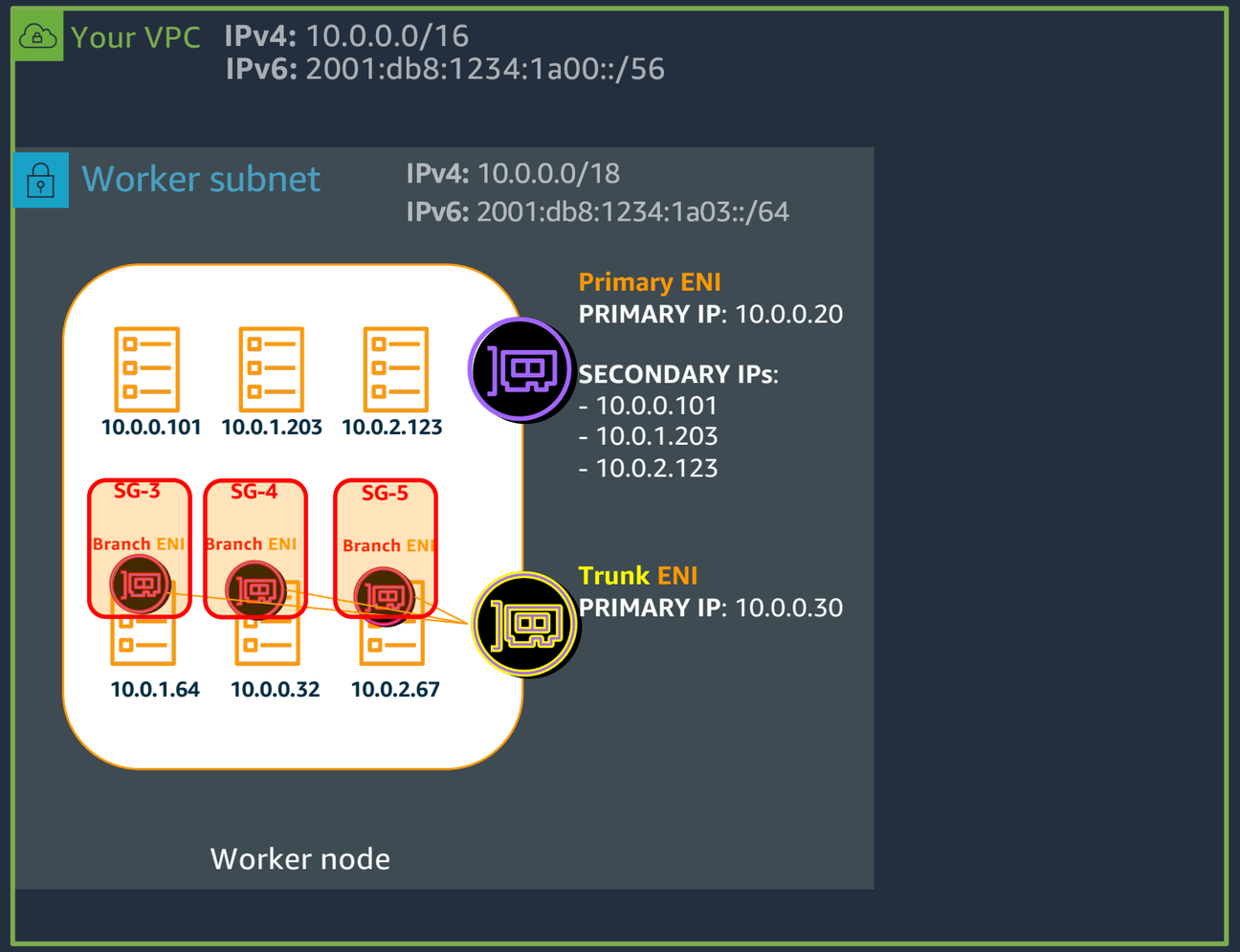
Branch and trunk ENIs



Feature only available through Amazon EKS or Amazon ECS and not directly on Amazon EC2

aws

Pod networking – Security groups for Pods



Each Pod gets a dedicated ENI (branch-ENI) mapped to a trunk ENI, allowing for independent security group configurations per Pod (`ENABLE_POD_ENI=true`)

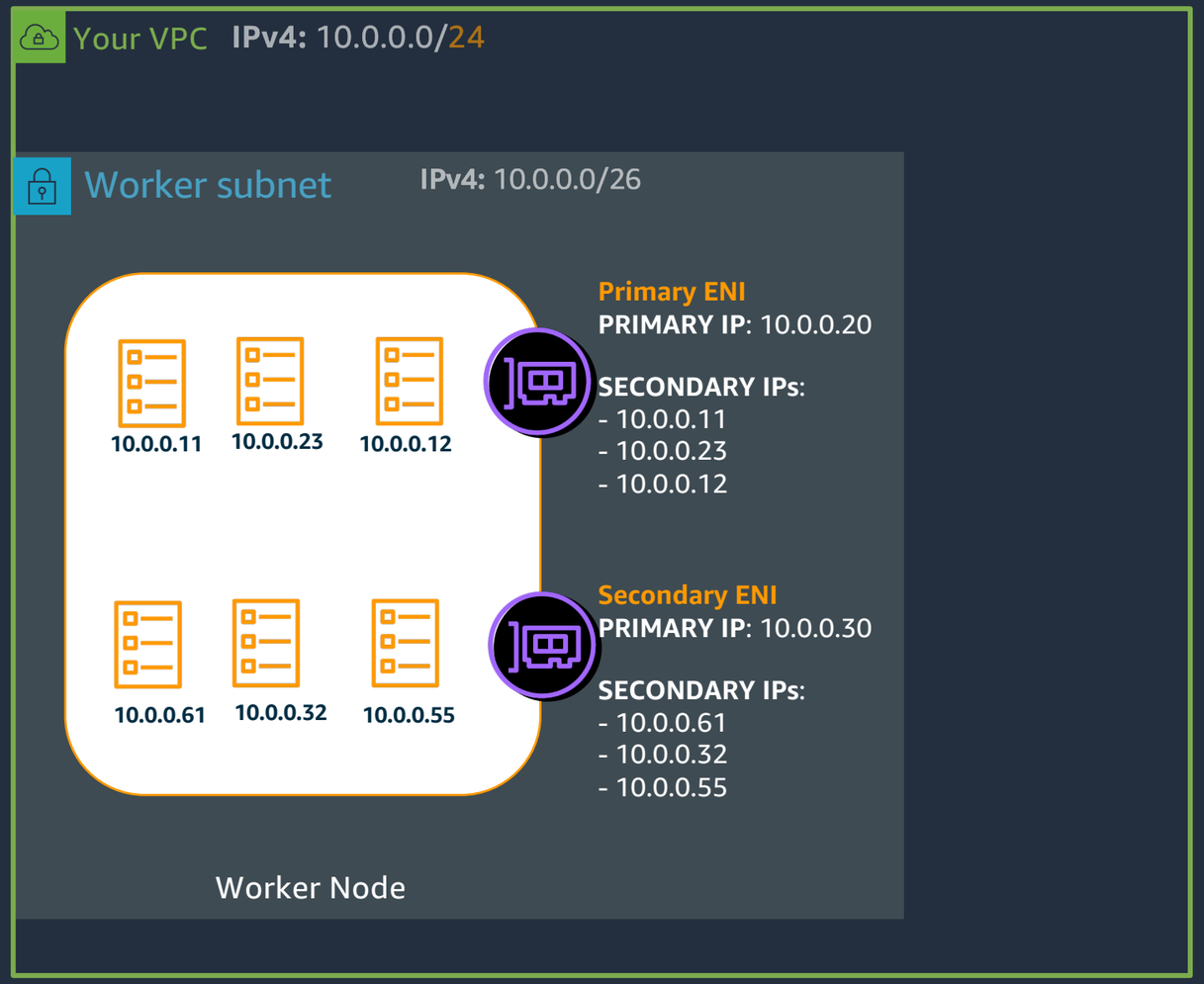
Considerations

- Supported by most Nitro instances
- SNAT is disabled for Pods using branch ENIs
- Cluster version 1.18+
- Review integration with NodePort and LoadBalancer services
- vpc.amazonaws.com/has-trunk-attached=true
- DISABLE_TCP_EARLY_DEMUX=true
- Branch ENI Limits
  - https://docs.aws.amazon.com/ko_kr/AmazonECS/latest/developerguide/container-instance-eni.html

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```


aws

Pod networking – Custom networking

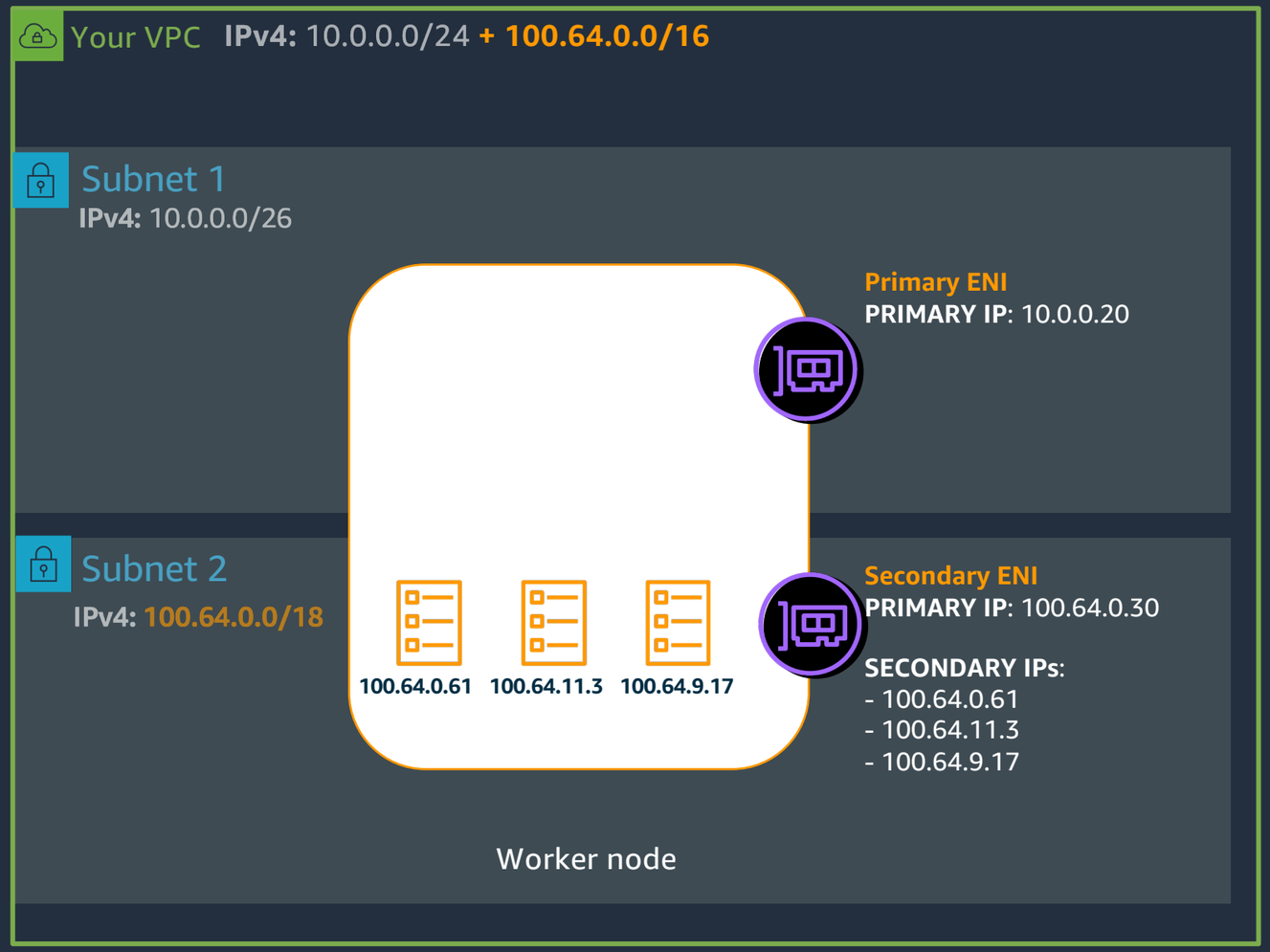


If the private IPv4 space is limited, the number of available IP addresses could constrain the number of Pods

/24 VPC CIDR provides 251 unique IPv4 addresses

aws

Pod networking – Custom networking



VPCs can have multiple IPv4 CIDR ranges

100.64.0.0/10 (RFC 6598) can be used in private networks

Worker node primary ENI is not used for Pods

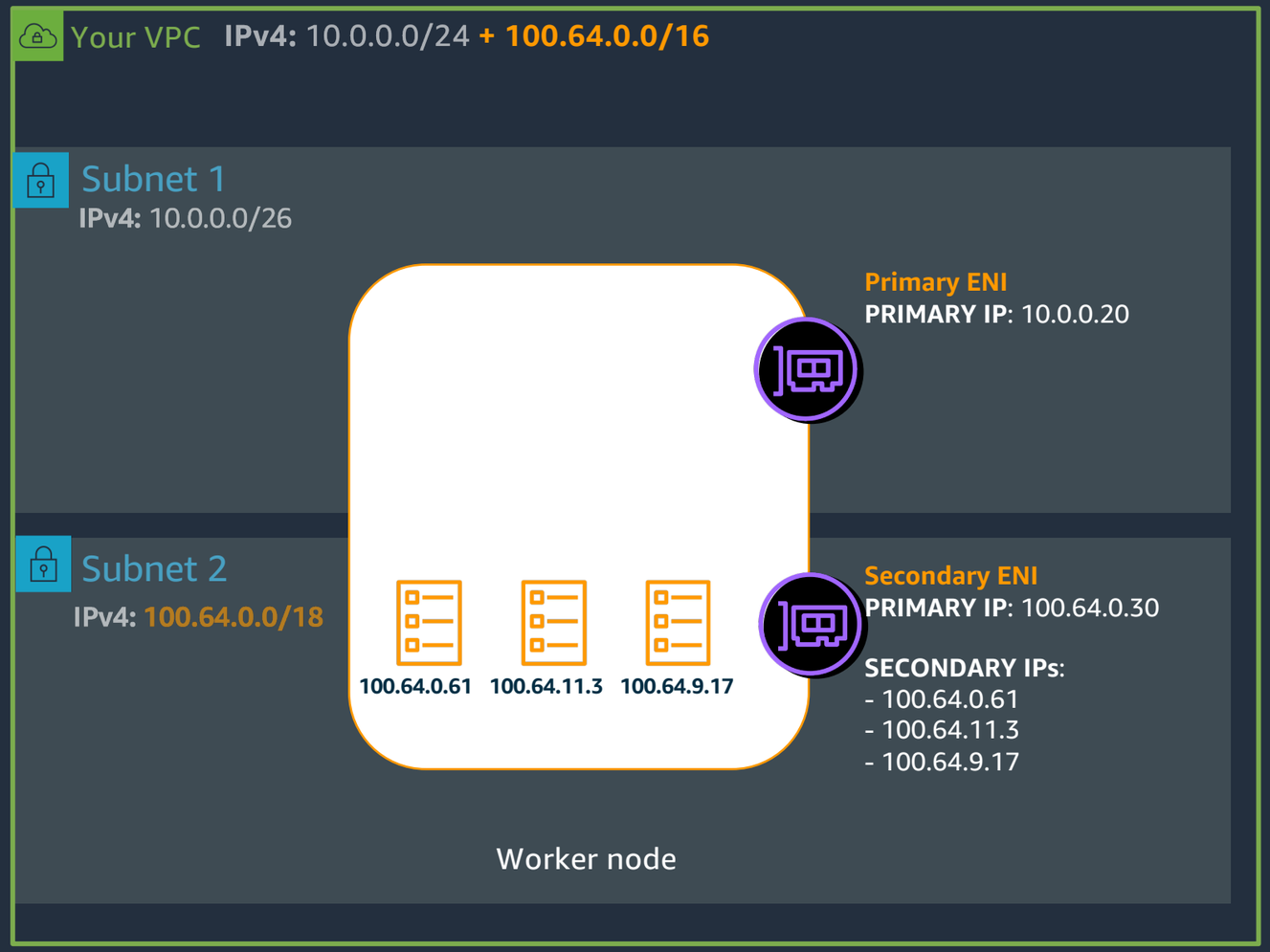
Custom networking can be combined with SNAT and prefix delegation

```
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: us-west-2a
spec:
  securityGroups:
    - sg-0dff111a1d11c1c11
  subnet: subnet-011b111c1f11fdf11
```

```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```


aws

Pod networking – Custom networking



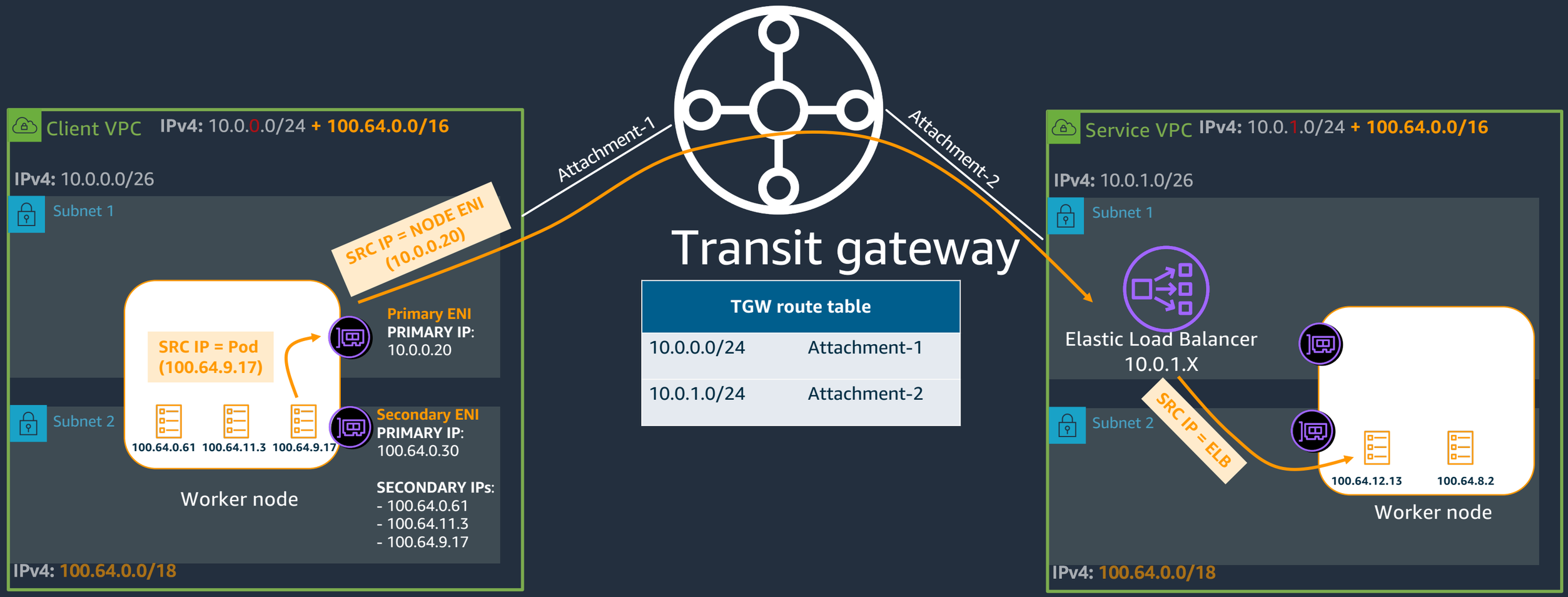
Kubelet BootstrapArguments parameter

Change "--max-pod=29" to "--max-pod=110"

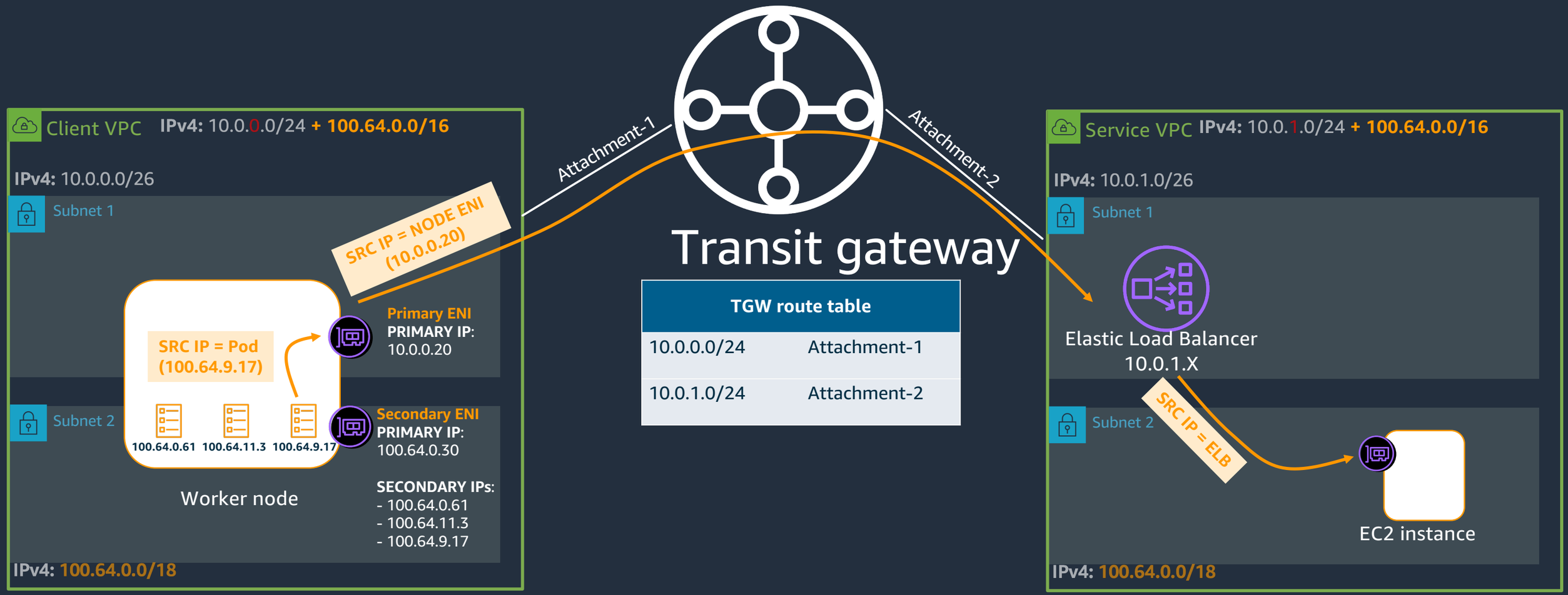
```
--use-max-pods false --kubelet-extra-args '--max-pods=110'
```


aws

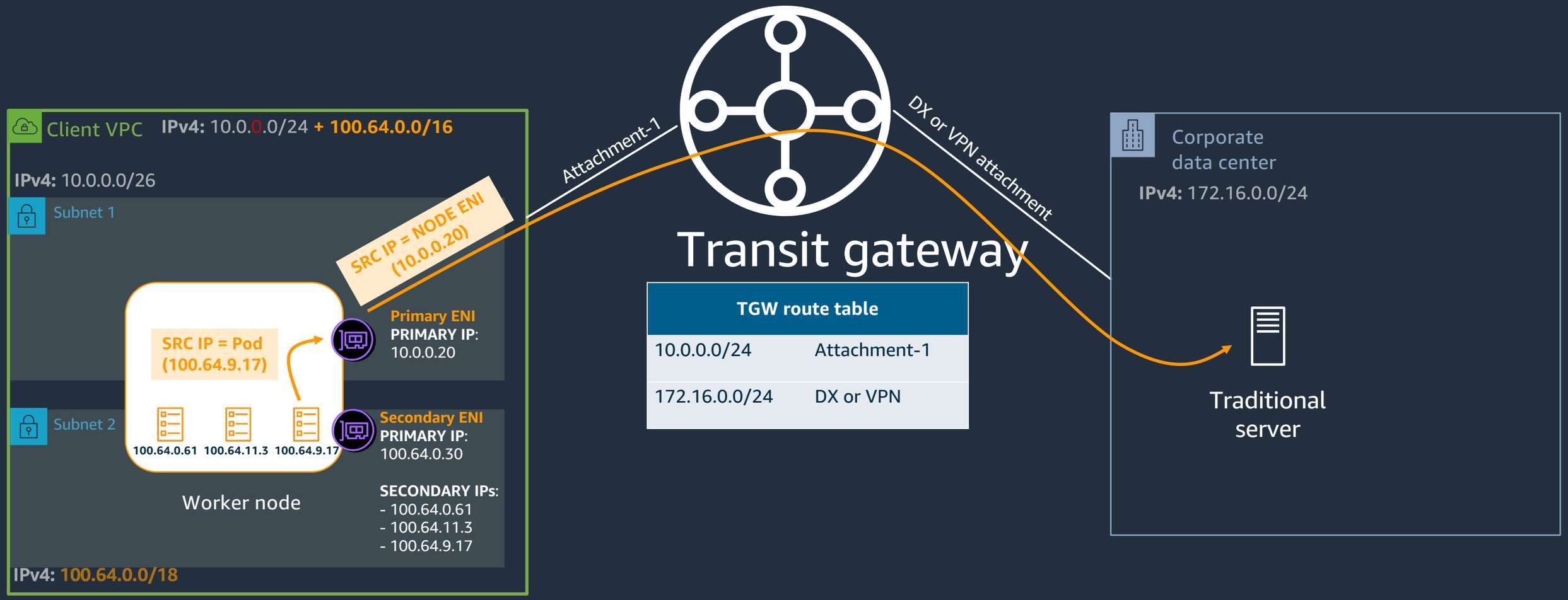
Custom networking + SNAT + transit gateway



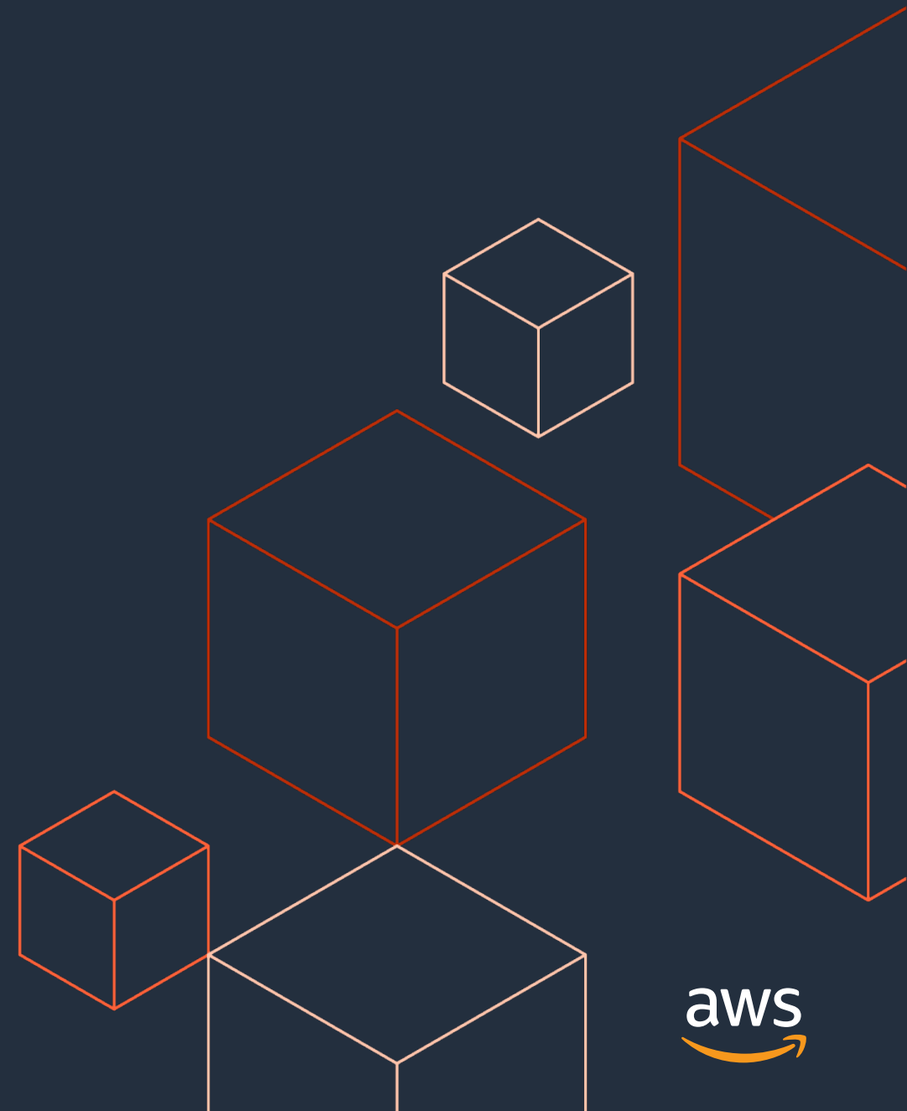
Custom networking + SNAT + transit gateway



Custom networking + SNAT + transit gateway

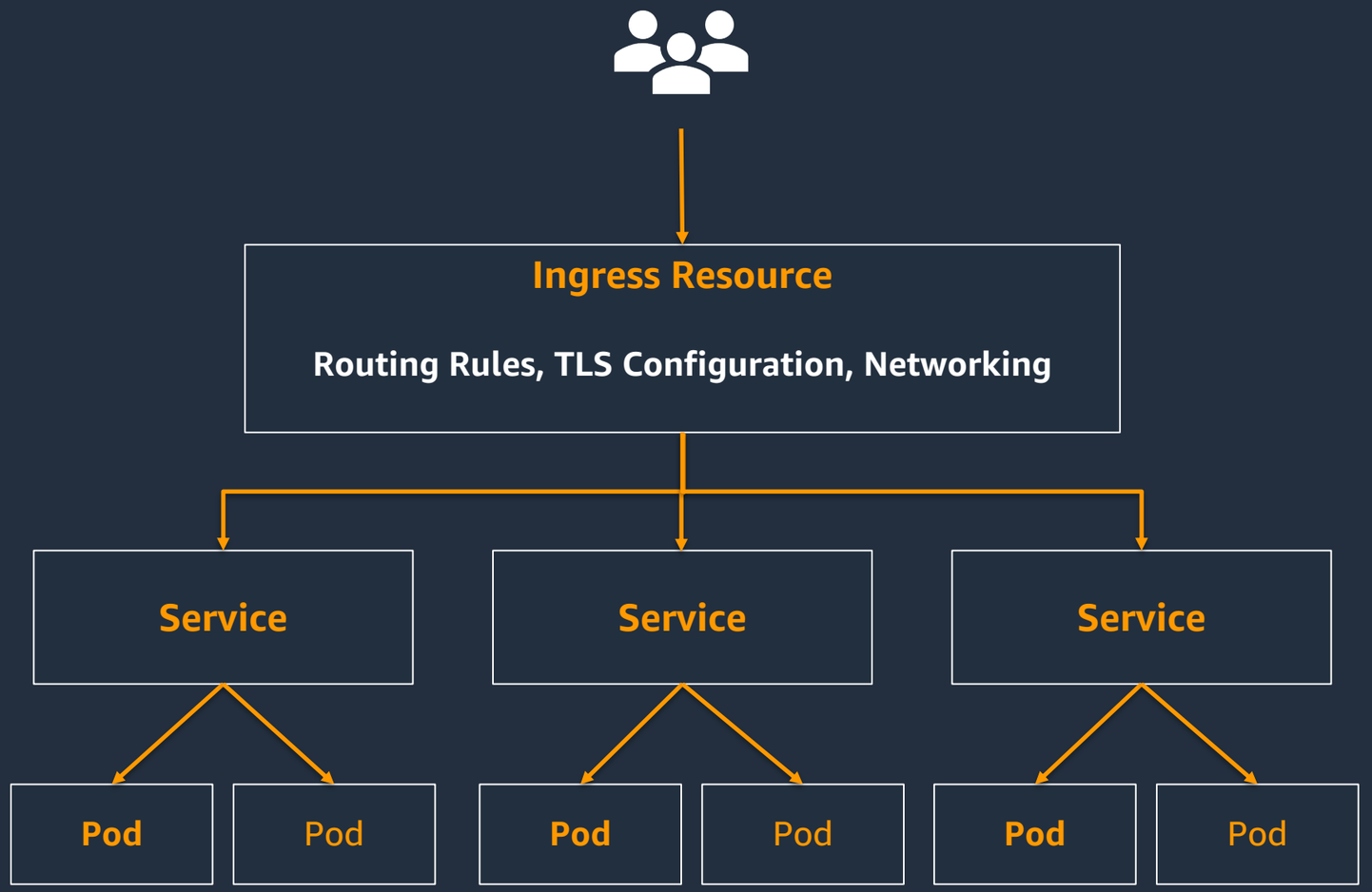


Ingress Controller



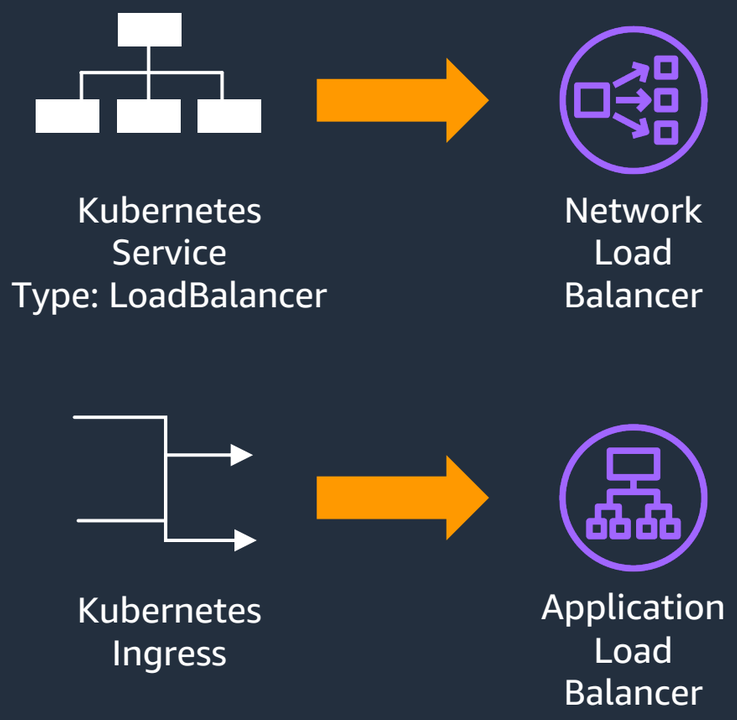
aws

Kubernetes Ingress



aws

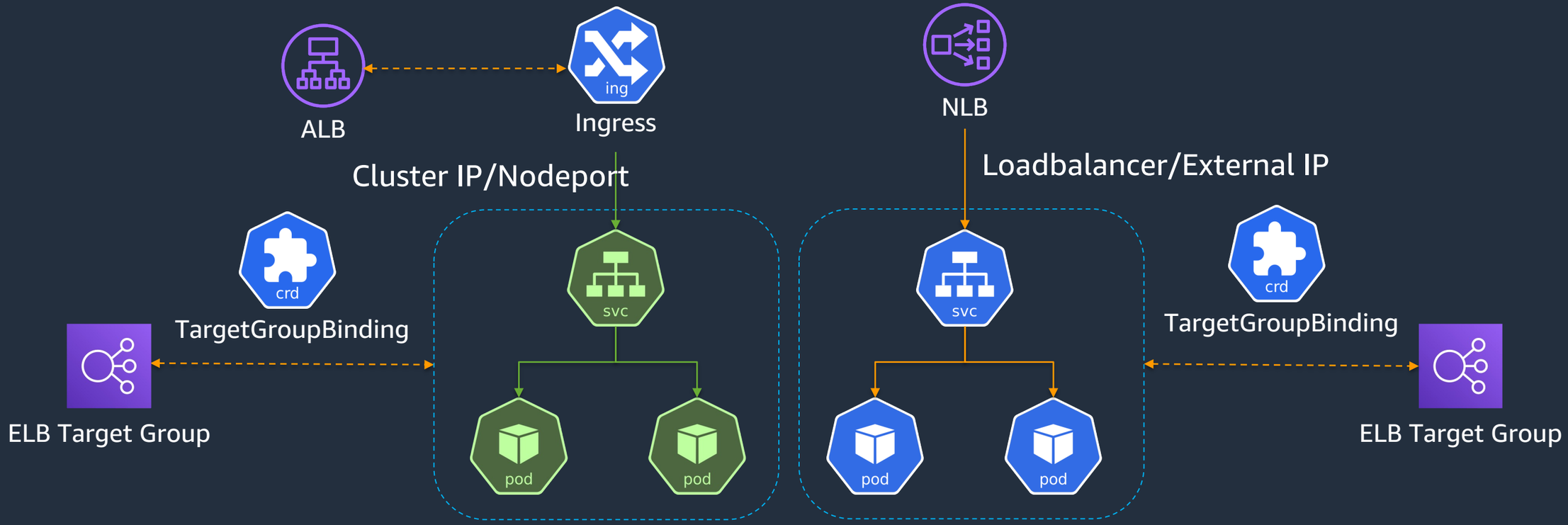
AWS Load Balancer Controller



- Manages AWS Elastic Load Balancing for an Amazon EKS or a Kubernetes cluster
- Uses standard Kubernetes resources
  - v1: Service (Type: LoadBalancer)
  - networking.k8s.io/v1: Ingress
  - networking.k8s.io/v1: IngressClass
- Custom resources
  - elbv2.k8s.aws/v1beta1: TargetGroupBinding
- Supports latest ELB features

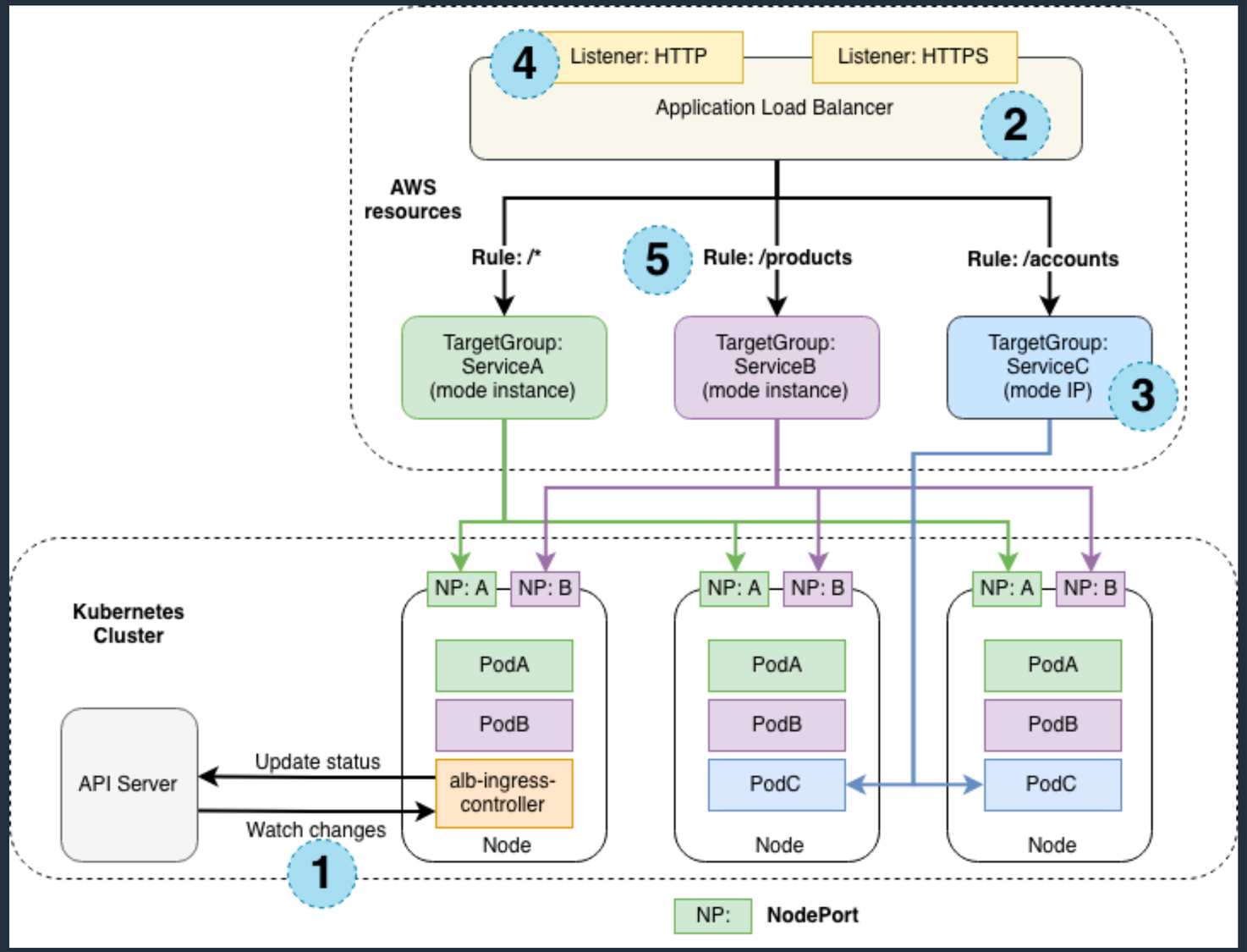
aws

AWS Load Balancer Controller

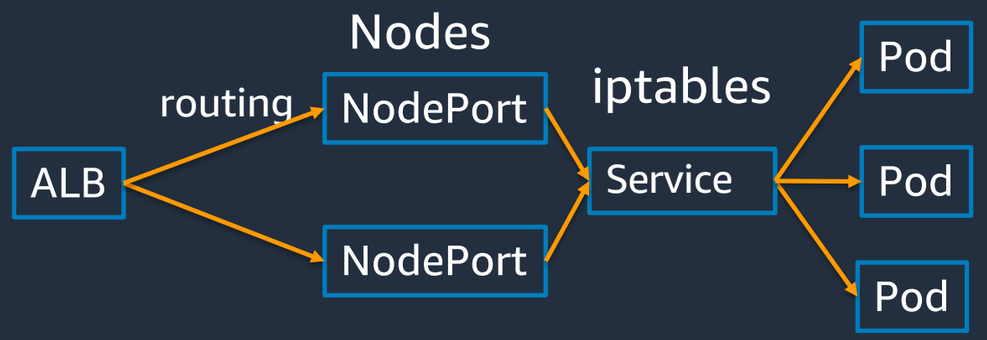


https://github.com/kubernetes-sigs/aws-load-balancer-controller

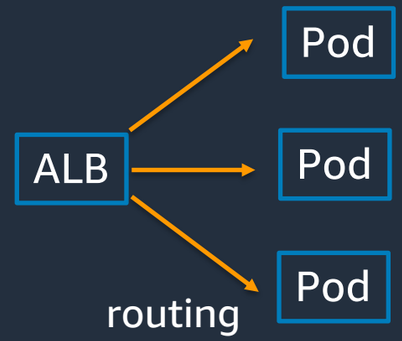
Ingress with AWS CNI



1. Instance Mode



2. IP Mode



https://github.com/kubernetes-sigs/aws-load-balancer-controller

Network Load Balancer provisioning in EKS

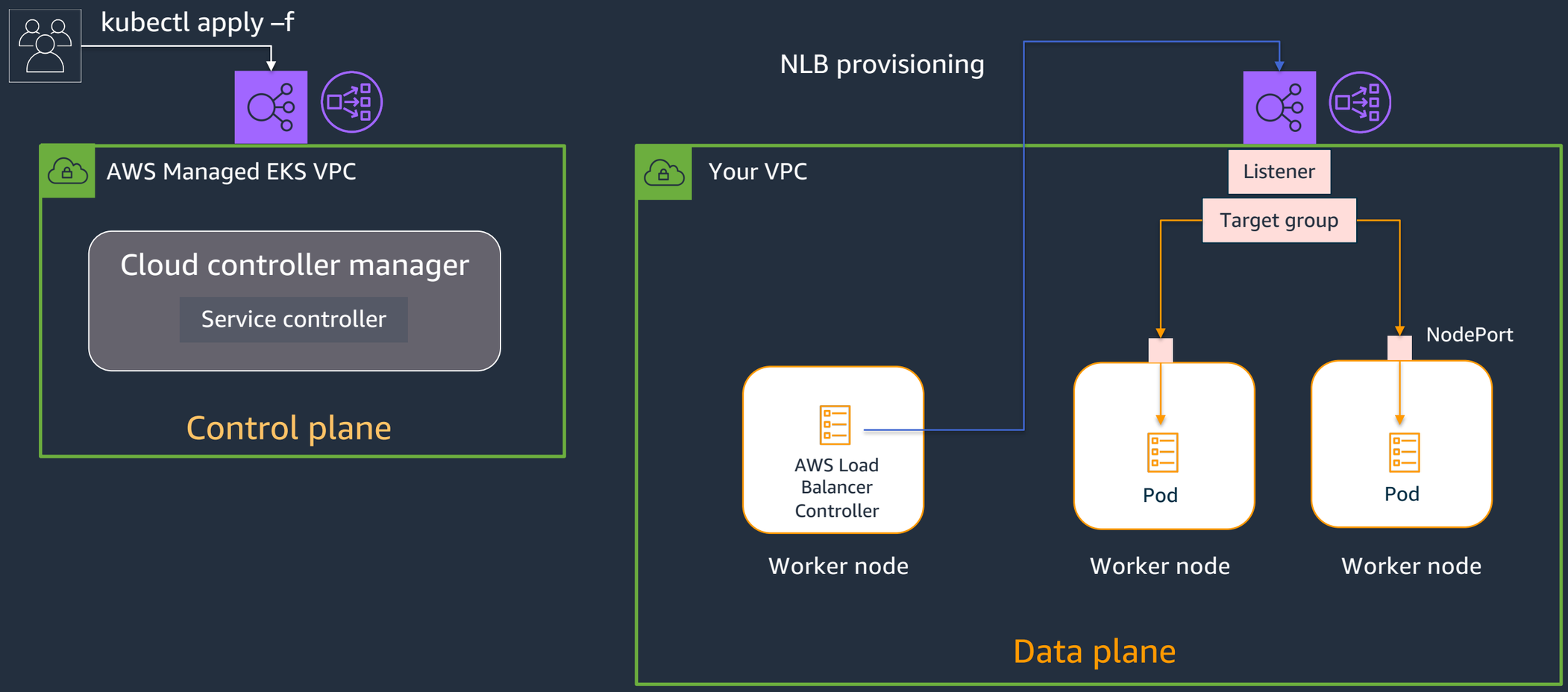
```
kind: Service
apiVersion: v1
metadata:
   name: nginx-service
   namespace: ingress-nginx
   annotations:
      service.beta.kubernetes.io/aws-load-balancer-backend-protocol: tcp
      service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: “true”
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
      service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: “instance"
      service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
   type: LoadBalancer
   selector:
      app: nginx
   ports:
      - name: http
        protocol: TCP
        port: 8080
        targetPort: 80
```

Using AWS Load Balancer Controller

requires version 2.2.0 or later

aws

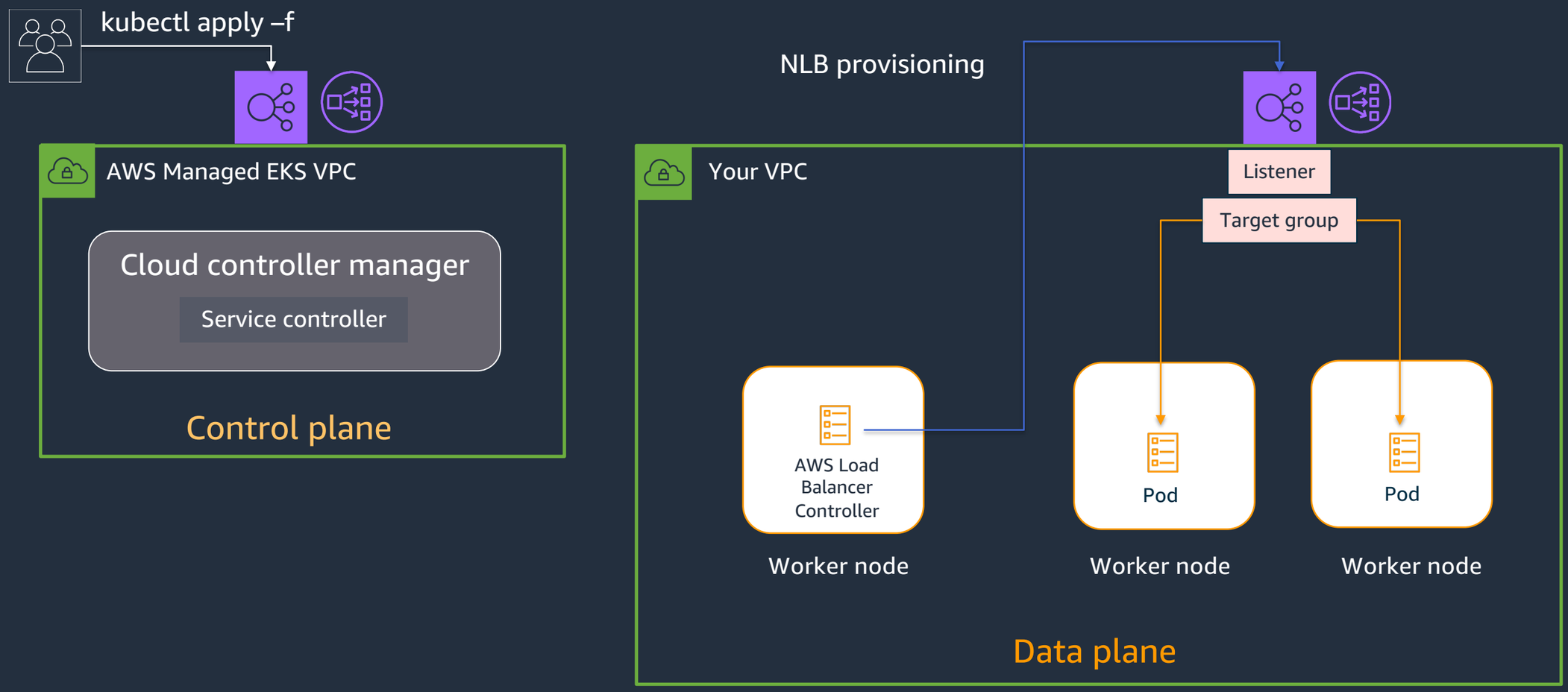
Network Load Balancer provisioning in EKS



service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: instance

aws

Network Load Balancer provisioning in EKS



service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip

aws

Application Load Balancer provisioning in EKS

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: echoserver-ingress
  namespace: echoserver
  annotations:
    alb.ingress.kubernetes.io/security-groups: sg-010fc3455c73f0a58, sg-049e999c68a291976
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/scheme: internet-facing

    kubernetes.io/ingress.class: alb
spec:

 rules:
    - host: echoserver.example.com
      http:
      paths:
        - path: /*
          backend:
            serviceName: echoserver-svc
            servicePort: 80
```


aws

Application Load Balancer provisioning in EKS

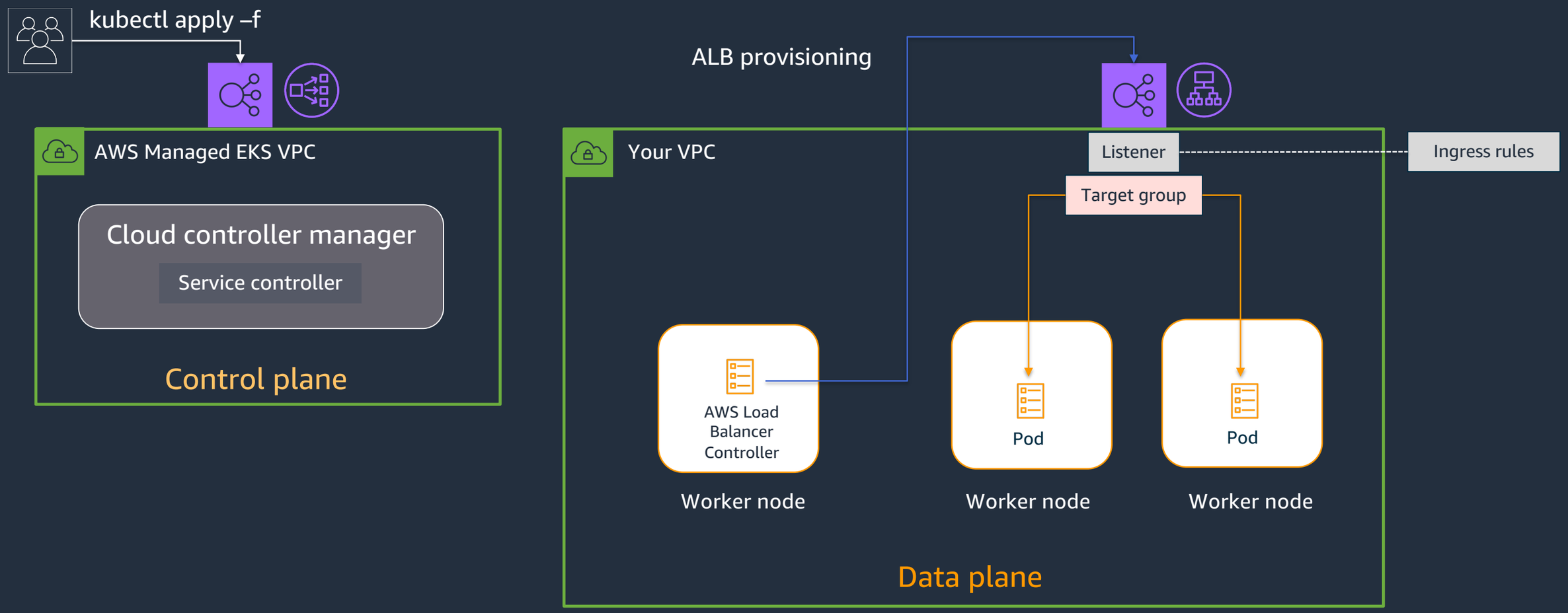
```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: echoserver-ingress
  namespace: echoserver
  annotations:
    alb.ingress.kubernetes.io/security-groups: sg-010fc3455c73f0a58, sg-049e999c68a291976
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/scheme: internet-facing


spec:
  ingressClassName: alb-ingress-class
  rules:
    - host: echoserver.example.com
      http:
      paths:
        - path: /*
          backend:
            serviceName: echoserver-svc
            servicePort: 80
```

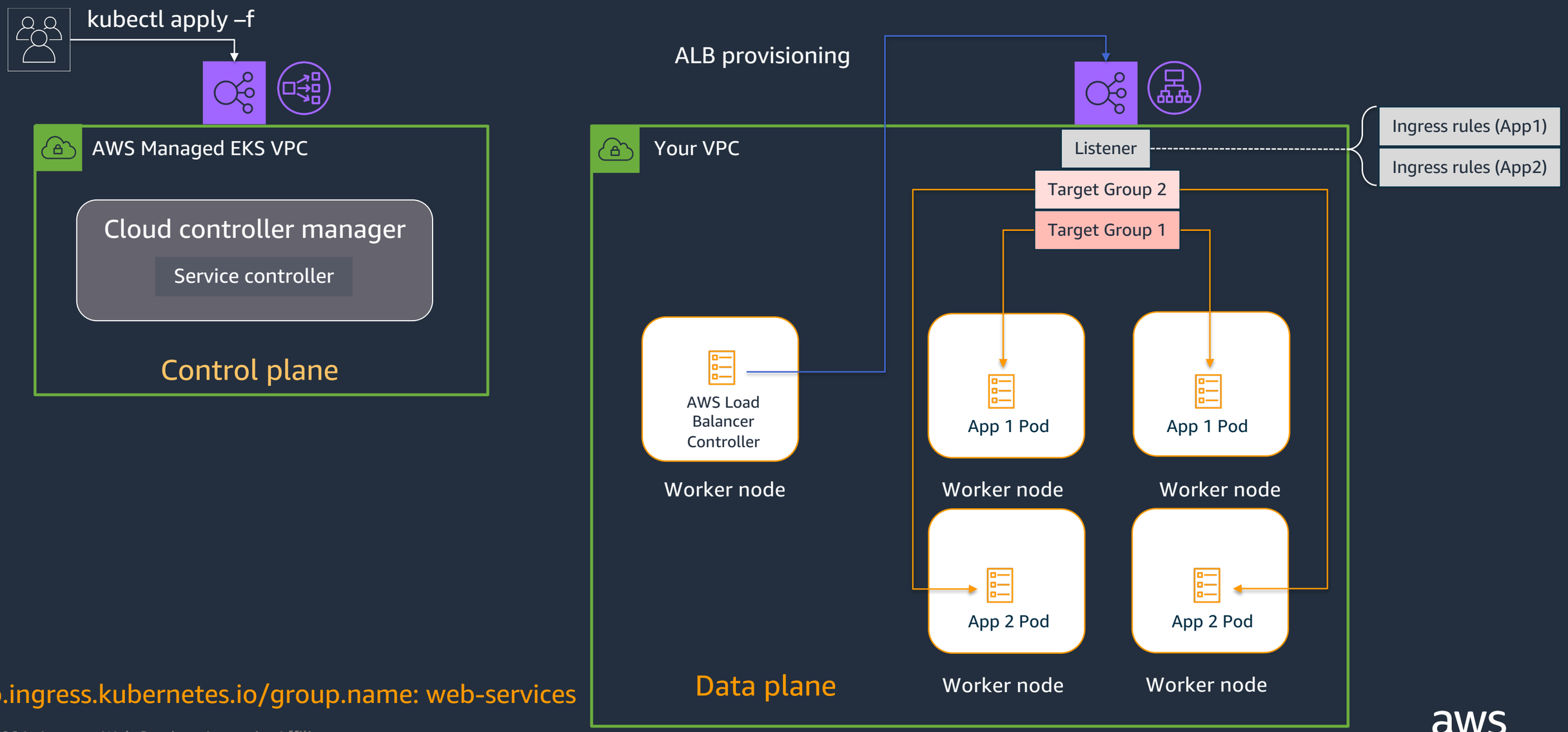
```
apiVersion: networking.k8s.io/v1beta1
kind: IngressClass
metadata:
  name: alb-ingress-class
spec:
  controller: ingress.k8s.aws/alb
```


aws

Application Load Balancer provisioning in EKS

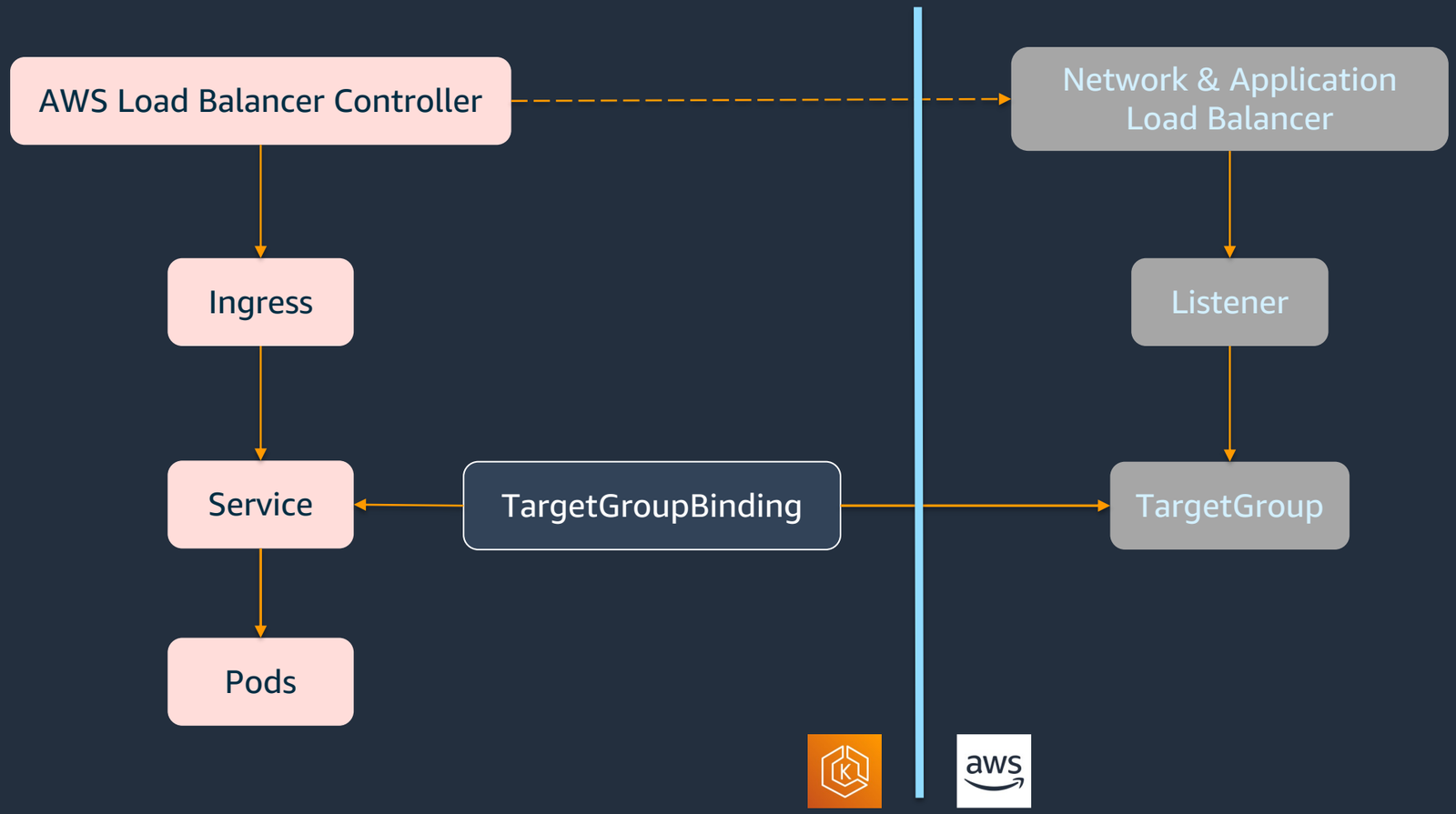


Application Load Balancer provisioning in EKS



alb.ingress.kubernetes.io/group.name: web-services

Binding AWS target groups and Kubernetes services



Workshop Links

- https://www.eksworkshop.com/docs/networking/prefix/
- https://www.eksworkshop.com/docs/networking/custom-networking/
- https://www.eksworkshop.com/docs/networking/security-groups-for-pods/
- https://www.eksworkshop.com/docs/fundamentals/exposing/

aws

Questions